

EXPERT INSIGHT

---

# Workflow Automation with Microsoft Power Automate

Use business process automation to achieve  
digital transformation with minimal code

**Second Edition**

**Aaron Guilmette**

**<packt>**

# Workflow Automation with Microsoft Power Automate

Second Edition

Use business process automation to achieve digital transformation with  
minimal code

**Aaron Guilmette**



BIRMINGHAM—MUMBAI

# Workflow Automation with Microsoft Power Automate

## Second Edition

Copyright © 2022 Packt Publishing

*All rights reserved.* No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

**Senior Publishing Product Manager:** Ashitosh Gupta

**Acquisition Editor – Peer Reviews:** Saby Dsilva

**Project Editor:** Meenakshi Vijay

**Content Development Editor:** Matthew Davies

**Copy Editor:** Safis Editing

**Technical Editor:** Srishty Bhardwaj

**Proofreader:** Safis Editing

**Indexer:** Tejal Daruwale Soni

**Presentation Designer:** Ganesh Bhadwalkar

First published: September 2020

Second edition: August 2022

Production reference: 2200223

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80323-767-1

[www.packt.com](http://www.packt.com)

# Contributors

## About the author

**Aaron Guilmette** is a Senior Program Manager at Microsoft, helping customers adopt the Microsoft 365 platform. He primarily focuses on collaborative and automation technologies, including Microsoft Teams, Exchange Online, Power Automate, and Azure Active Directory. Aaron lives in the Detroit, Michigan area with his five children. When he's not busy solving technical problems, writing, or running his children to events, he's likely making a pizza.

*I want to dedicate this book to my kids: Liberty, Hudson, Glory, Anderson, and Victory. Without them, my life would be too quiet. I especially thank my partner, Christine, for continuing to support me throughout my career. If anyone deserves an award, it's her. I also want to thank everyone in the Power Automate community who has endeavored to make us more productive at work so we can be more present with our families.*



## About the reviewers

**Benedikt Bergmann** is a Solutions Architect and Developer working for CRM Konsulterna in Stockholm, Sweden. He is originally from Germany, but moved to Sweden in 2017 with his wife.

Benedikt has his roots as a .NET developer with some front-end elements. He has worked within Dynamics and Dataverse since 2014 and now implements solutions based on Dynamics and the Power Platform. Benedikt shares his knowledge actively with the community through his blog, YouTube channel, and presentations. At the beginning of 2021, Benedikt was recognized as a Microsoft Most Valuable Professional in Business Applications.

**Mohamed Abdel Ghaffar** is a Solutions Architect with over 14 years' experience with Microsoft technologies and platforms (including web development, integration architecture, big data analytics, and the Power Platform). He has architected, designed, and developed enterprise solutions for governmental, banking, and telecommunication entities in Egypt and the Gulf region. Mohamed has achieved certifications related to the cloud, data analytics, and the Power Platform. In his free time, Mohamed likes to play chess and football and spend quality time with his family.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>





# Table of Contents

<b>Preface</b>	<b>xvii</b>
<hr/>	
<b>Chapter 1: Microsoft Power Automate</b>	<b>1</b>
<hr/>	
What is Power Automate? .....	2
Choosing what to automate .....	3
Reviewing general terminology .....	5
Business process • 5	
Workflow • 6	
REST • 6	
Learning Power Automate terminology .....	6
Flow • 7	
Connectors • 8	
Triggers • 9	
Steps • 9	
Actions • 9	
Conditions • 10	
Dataverse • 10	
Gateways • 10	
Templates • 11	
Packages • 12	
Branching • 12	
Getting Power Automate .....	13
Summary .....	13

---

<b>Chapter 2: Getting Started with Power Automate</b>	<b>15</b>
<b>Logging in to Power Automate</b> .....	<b>15</b>
End user web portal interface • 16	
Mobile app interface • 18	
Power Automate Desktop interface • 20	
Admin interface • 21	
<b>Creating your first flow</b> .....	<b>22</b>
Understanding the flow components • 22	
Creating and executing the flow • 23	
Examining the flow • 29	
<b>Summary</b> .....	<b>30</b>
 <b>Chapter 3: Working with Email</b>	 <b>31</b>
<b>Learning about email connectors and actions</b> .....	<b>32</b>
<b>Working with email</b> .....	<b>33</b>
Reading email • 33	
Handling attachments • 36	
<i>Creating the flow</i> • 37	
<i>Verifying completion</i> • 44	
Sending email • 46	
<b>Summary</b> .....	<b>49</b>
 <b>Chapter 4: Copying Files</b>	 <b>51</b>
<b>Learning about file connectors and actions</b> .....	<b>51</b>
<b>Working with files</b> .....	<b>53</b>
Copying files to SharePoint • 54	
<i>Verifying results with Test Flow</i> • 61	
<i>Verifying the results manually</i> • 62	
Publishing files to Dropbox • 63	
<i>Verifying the results</i> • 66	
<b>Summary</b> .....	<b>68</b>

<b>Chapter 5: Creating Button Flows</b>	<b>69</b>
Learning about button flows .....	69
Creating a button flow to email a manager .....	71
Executing a button flow .....	79
Summary .....	82
<b>Chapter 6: Generating Push Notifications</b>	<b>83</b>
Learning about push notifications .....	83
Configuring a notification for emails from your manager .....	85
Introducing conditions • 86	
Creating the flow • 88	
Reviewing a notification .....	94
Reworking the flow .....	95
Summary .....	96
<b>Chapter 7: Working with Shared Flows</b>	<b>97</b>
Understanding shared flows .....	97
Sharing a cloud flow .....	98
Sharing a desktop flow .....	102
Creating a basic desktop flow • 102	
Sharing a desktop flow • 106	
Sharing a flow with run only permissions .....	107
Managing shared flows .....	111
Summary .....	112
<b>Chapter 8: Working with Conditions</b>	<b>115</b>
Understanding condition operators .....	116
Evaluating objects • 118	
Using expressions and multiple conditions .....	119
Adding multiple conditions • 119	
Adding condition groups • 122	

Using a switch condition • 127	
<b>Summary .....</b>	<b>133</b>
<b>Chapter 9: Getting Started with Approvals</b>	<b>135</b>
<b>Understanding Dataverse .....</b>	<b>135</b>
<b>Creating an approval flow .....</b>	<b>137</b>
Creating a SharePoint site and list • 138	
Creating an approval • 141	
Adding calendar events and notifications • 149	
Starting the flow • 152	
<b>Responding to approvals .....</b>	<b>153</b>
<b>Summary .....</b>	<b>156</b>
<b>Chapter 10: Working with Multiple Approvals</b>	<b>157</b>
<b>Working with sequential approvals .....</b>	<b>157</b>
<b>Working with parallel approvals .....</b>	<b>158</b>
<b>Working with advanced scenarios .....</b>	<b>159</b>
Mixed approval types • 159	
Everyone must approve • 159	
<b>Creating a basic sequential approval .....</b>	<b>159</b>
Configuring the prerequisites • 160	
Creating the flow • 161	
<i>Creating the trigger • 161</i>	
<i>Creating the first-stage approval • 162</i>	
<i>Creating the second-stage approval • 168</i>	
<i>Completing the flow • 170</i>	
Testing the flow • 173	
<b>Adding parallel branches .....</b>	<b>174</b>
<b>Summary .....</b>	<b>183</b>

<b>Chapter 11: Posting Approvals to Teams</b>	<b>185</b>
Understanding the flow .....	185
Configuring prerequisites .....	186
A SharePoint site • 186	
Power Automate app for Teams • 187	
Configuring an approval flow to use Teams .....	188
Getting the requester's information • 189	
Creating the approval • 192	
Returning the response • 197	
Testing the flow .....	204
Requesting approval • 206	
Approving the request • 206	
Reviewing the response • 207	
Summary .....	208
<b>Chapter 12: Using a Database</b>	<b>209</b>
Understanding database connectors, triggers, and actions .....	210
Triggers • 210	
Actions • 211	
Connecting to a database .....	211
Creating a server • 211	
Creating a database • 215	
Creating a database table • 217	
Creating a connection to a database .....	220
Adding content to a database .....	222
Creating the flow • 223	
Executing the flow • 226	
Verifying the flow • 226	
<i>Reviewing the run history • 226</i>	
<i>Reviewing the SQL data • 227</i>	
Summary .....	229



<b>Chapter 13: Working with Microsoft Forms</b>	<b>231</b>
Understanding the Forms connector triggers and actions .....	231
Triggers • 232	
Actions • 232	
Creating a basic form .....	232
Processing a form with Power Automate .....	236
Testing the flow .....	242
Verifying the result .....	243
Reviewing the run history • 243	
Reviewing the SQL data • 244	
Summary .....	245
<b>Chapter 14: Accepting User Input</b>	<b>247</b>
Understanding the user input options .....	247
Creating a flow that uses all input types .....	250
Configuring the prerequisites • 250	
Creating the flow • 252	
Executing the flow • 259	
Verifying the flow execution • 262	
Expanding further .....	263
Summary .....	267
<b>Chapter 15: Automating Azure AD</b>	<b>269</b>
Technical requirements .....	270
Learning about connectors and actions .....	270
HTTP action • 270	
Parse JSON • 272	
Create CSV table • 272	
Configuring prerequisites .....	272
Configuring an Azure AD application • 272	

---

Creating a client secret • 278	
<b>Creating an HTTP flow .....</b>	<b>281</b>
Gathering the data • 281	
Parsing the output • 288	
Creating a report • 290	
<b>Testing the flow .....</b>	<b>296</b>
<b>Verifying the flow .....</b>	<b>298</b>
<b>Expanding further .....</b>	<b>299</b>
<b>Summary .....</b>	<b>299</b>
 <b>Chapter 16: Introducing Robotic Process Automation .....</b>	 <b>301</b>
<hr/>	
<b>Technical requirements .....</b>	<b>302</b>
<b>Learning about actions and variable passing .....</b>	<b>303</b>
Run a flow built with Power Automate for desktop action • 303	
Variable passing • 303	
The Power Automate recorder • 304	
<b>Configuring prerequisites .....</b>	<b>304</b>
Configuring Power Automate Desktop • 304	
Configuring a Microsoft Access database • 305	
Configuring the Microsoft on-premises data gateway • 308	
<b>Configuring an RPA flow .....</b>	<b>311</b>
Configuring the flow framework • 312	
Configuring an application to open • 316	
Running the Power Automate recorder • 319	
Updating the variables • 323	
Finishing the flow • 327	
<b>Testing the flow .....</b>	<b>328</b>
<b>Verifying the flow .....</b>	<b>330</b>
<b>Summary .....</b>	<b>331</b>

<b>Chapter 17: Introducing AI Models</b>	<b>333</b>
Learning about AI models .....	334
Technical requirements .....	335
Configuring prerequisites .....	338
Creating a Forms survey • 338	
Creating an Excel workbook • 340	
Creating a sentiment analysis flow .....	342
Testing the flow .....	352
Reviewing the output .....	353
Expanding further .....	355
Summary .....	355
<b>Chapter 18: Exporting, Importing, and Distributing Flows</b>	<b>357</b>
Exporting a flow .....	357
Importing a flow .....	361
Distributing a flow .....	365
Sending a flow via Power Automate • 365	
Publishing a flow to the template gallery • 368	
Summary .....	370
<b>Chapter 19: Monitoring and Troubleshooting Flows</b>	<b>373</b>
Monitoring flows .....	373
Viewing the run history • 374	
Resubmitting a flow • 376	
Reviewing email error reports .....	378
Resolving authentication errors .....	379
Examining detailed errors with the flow checker .....	380
Understanding error codes .....	381
Finding additional resources .....	383
Summary .....	384
Why subscribe? .....	385

<b>Other Books You May Enjoy</b>	<b>387</b>
<b>Index</b>	<b>391</b>



# Preface

Microsoft Power Automate is a business productivity tool designed to bring automation capabilities to both cloud and on-premises applications. The ubiquity and proliferation of cloud-based software-as-a-service technologies has resulted in organizations using a myriad of disconnected tools as part of their business operations.

Throughout this updated and revised book, you'll learn how Power Automate can be used to drive efficiency in automation by connecting all these disconnected parts and help you reduce the time spent performing repetitive tasks.

## Who is this book for?

This book is for individuals who are new to the Microsoft Power Platform (including Power Automate, Power Apps, and Power BI). Readers can have little to no experience in coding, automation, or software development practices. This book will walk readers through very basic tasks to gain an understanding of the tools and leave them with completed workflows that can be exported and used in their daily work or further refined and adapted to more complex processes.

## What this book covers

*Chapter 1, Introducing Microsoft Power Automate*, introduces the basic concepts of workflows and automation.

*Chapter 2, Getting Started with Power Automate*, dives into the Power Automate interface and application components.

*Chapter 3, Working with Email*, explains how to configure Power Automate to process and interact with emails and attachments.

*Chapter 4, Copying Files*, explores how to use Power Automate to copy files between cloud-based services.

*Chapter 5, Creating Button Flows*, demonstrates how to use button flows (also called manual or instant flows) and shows how they can be executed from mobile devices.

*Chapter 6, Generating Push Notifications*, explains how to configure flows with push notifications to mobile devices.

*Chapter 7, Working with Shared Flows*, covers how to share and co-manage flows with your peers.

*Chapter 8, Working with Conditions*, explains how to enable multiple outcomes for flows based on criteria evaluation.

*Chapter 9, Getting Started with Approvals*, discusses how to use approval workflows for documents.

*Chapter 10, Working with Multiple Approvals*, builds on the knowledge gained in the previous chapter and expands it using conditions, branching, and sequential approvals.

*Chapter 11, Posting Approvals to Teams*, demonstrates how to integrate approval workflows into Microsoft Teams.

*Chapter 12, Using a Database*, covers connecting to a database and using it to store or retrieve information as part of a flow.

*Chapter 13, Working with Microsoft Forms*, demonstrates how to use Power Automate to process response data from Microsoft Forms.

*Chapter 14, Accepting User Input*, explores configuring a button flow to accept a variety of user input types.

*Chapter 15, Automating Azure AD*, introduces connectors for working with Microsoft Graph.

*Chapter 16, Introducing Robotic Process Automation*, demonstrates capturing screen, keyboard, and mouse data to automate legacy desktop applications.

*Chapter 17, Introducing AI Models*, introduces the Sentiment Analysis AI model for evaluating text.

*Chapter 18, Exporting, Importing, and Distributing Flows*, discusses how to import, export, and transport flows between environments.

*Chapter 19, Monitoring and Troubleshooting Flows*, contains troubleshooting tips for when things go wrong.

## To get the most out of this book

To get the most out of this book, you should have a Microsoft 365 subscription with access to a premium Power Automate plan to enable access to premium connectors, AI models, and Robotic Process Automation.

You should also have an internet-connected computer with a modern browser such as Microsoft Edge or Google Chrome. You can obtain a free Microsoft 365 subscription from <https://aka.ms/e5trial>.

For *Chapter 2, Getting Started with Power Automate*, and *Chapter 4, Copying Files*, you will need accounts on Twitter and Dropbox, respectively.

For *Chapter 16, Introduction to Robotic Process Automation*, you will need a Power Automate license that includes unattended Robotic Process Automation.

For *Chapter 17, Introduction to AI Models*, you will need AI Builder capacity. The unattended Robotic Process Automation licensing includes a base level of AI Builder capacity suitable for completing the examples.

## Download the example code files

The code bundle for the book is hosted on GitHub at <https://github.com/PacktPublishing/Workflow-Automation-with-Microsoft-Power-Automate-2nd-Edition>. We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

## Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: [https://static.packt-cdn.com/downloads/9781803237671\\_ColorImages.pdf](https://static.packt-cdn.com/downloads/9781803237671_ColorImages.pdf).

## Conventions used

There are a number of text conventions used throughout this book.

**CodeInText:** Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. For example: “Update the prompt to say `Please enter today's date.`”

A block of code is set as follows:

```
{
  "type": "thumbnail",
  "fileName": "FavoriteThing.jpg",
  "fieldName": "Picture",
  "serverUrl": "https://learningpowerautomate.sharepoint.com/sites/
FavoriteThings",
```



```
"serverRelativeUrl": "@  
{decodeURIComponent(decodeURIComponent(outputs('Add_attachment')?['body/  
Id']))}",  
  "id": "@{guid()}"  
}
```

**Bold:** Indicates a new term, an important word, or words that you see on the screen. For instance, words in menus or dialog boxes appear in the text like this. For example: “A **gateway** is a software application installed on an on-premises computer.”



Warnings or important notes appear like this.



Tips and tricks appear like this.

## Get in touch

Feedback from our readers is always welcome.

**General feedback:** Email [feedback@packtpub.com](mailto:feedback@packtpub.com) and mention the book’s title in the subject of your message. If you have questions about any aspect of this book, please email us at [questions@packtpub.com](mailto:questions@packtpub.com).

**Errata:** Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you reported this to us. Please visit <http://www.packtpub.com/submit-errata>, click **Submit Errata**, and fill in the form.

**Piracy:** If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at [copyright@packtpub.com](mailto:copyright@packtpub.com) with a link to the material.

**If you are interested in becoming an author:** If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit <http://authors.packtpub.com>.

## Share your thoughts

Once you've read *Workflow Automation with Microsoft Power Automate, Second Edition*, we'd love to hear your thoughts! Please [click here](#) to go straight to the Amazon review page for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don't worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below



<https://packt.link/free-ebook/9781803237671>

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly



# 1

## Introducing Microsoft Power Automate

Business activities in the Information Age are filled with repetitive tasks: receive an email, generate a purchase order, send a message, route a document, approve a time-off request. In many cases, these activities don't generate real value, though they do need to get done to help support business goals.

Computers introduced the promise of helping us do more, but a lot of that has resulted in there being more to do in order to get the same value. What if we could use technology to handle routine tasks and save our strength for doing the things that require skill and thinking?

In this book, we're going to learn the basics of Microsoft Power Automate, a tool designed to help you automate repetitive tasks and get you back to generating value.

This chapter focuses on getting an understanding of some of the basic concepts of Power Automate:

- What is Power Automate?
- Choosing what to automate
- Reviewing general terminology
- Learning Power Automate terminology

By the end of this chapter, you will have been introduced to the components of Power Automate and when we might use them to streamline daily operations—from the basics of email management to more complex business operations.

## What is Power Automate?

Power Automate, part of the Power Platform family of products, is a workflow engine that can be used to automate common business processes or sequences based on conditions or scenarios. Power Automate (formerly known as Microsoft Flow) is primarily a web-based tool designed to interface with a growing library of software from both Microsoft and other vendors. Due to its no-code/low-code design, Power Automate can be approached by individuals with any technical skill level—including office or business users with no coding experience, system administrators, and programmers.



### Power Automate Desktop

In early 2021, Microsoft introduced Power Automate Desktop—a product that could be used to help automate processes on legacy computer applications using **Robotic Process Automation**. Microsoft has now made that Power Automate Desktop application a native part of Windows 11 as well!

Many readers may be familiar with the concept of SharePoint workflows. On the SharePoint platform, you can use products such as SharePoint Designer and Workflow Manager to kick off business processes based on work activities—such as a document being checked in to a particular library. One of the great things about SharePoint workflows is that they can automate business processes and tasks inside the SharePoint environment. However, one of the drawbacks of SharePoint workflows is that they *only* automate business processes and tasks inside that SharePoint environment.

While some vendors have created integration packages to tie SharePoint workflows to external products or have developed full replacements for SharePoint Designer and Workflow Manager, many of them are limited to interfacing with data inside SharePoint.

This is where the power of the Power Automate platform really shines—it has native connectivity to hundreds of applications out of the box and isn't dependent just on SharePoint. Not only does it have native integration for the world's most popular applications, but it's also extensible: you can develop your own connectivity solutions to work with your organization's custom apps.

Power Automate's capabilities are limited only by your imagination and the services offered by the applications you wish to integrate.

You can see that in the following diagram, a sample purchase order workflow ties together the SharePoint, Outlook, and Microsoft Approvals apps:

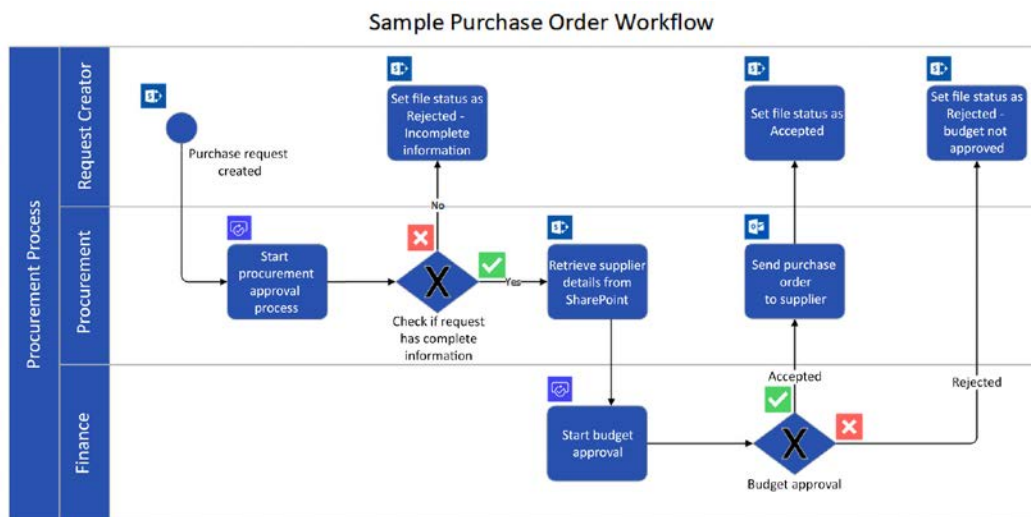


Figure 1.1: Sample purchase order workflow

This basic workflow, connecting three disparate applications, is just the beginning of what Power Automate can do for you.

## Choosing what to automate

Frequently, people do work that they don't need to do. It's important to differentiate between *work that doesn't need to be done at all* and *work that needs to be done, but doesn't require you to do it*.

From both the administrative and end user perspectives, there are a number of activities, processes, and tasks in Office 365 and other line-of-business applications that can be automated through the use of Power Automate.



Automation is a key business technology to reduce the impact of repetitive, low-skill tasks on the workforce. Consider the following diagram:

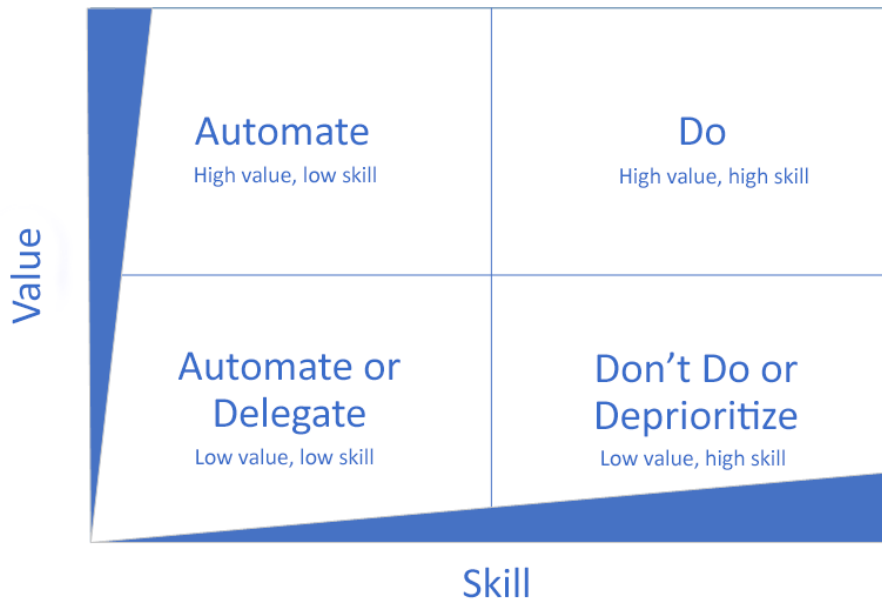


Figure 1.2: Candidates for automation

When looking at business processes, they can generally be divided into one of the four quadrants:

- **High value, low skill:** Requires minimal specialist skill, but is critical or produces high-value output.
- **High value, high skill:** Requires human intelligence or processing to determine value, skills, and relationships.
- **Low value, low skill:** Requires minimal specialist skill and also produces an output of small value.
- **Low value, high skill:** Requires a high degree of focus or skill but produces an output of small value.

As you examine tasks in your daily routine, you can evaluate them against the preceding matrix to help understand whether it is something that can or should be automated. Items that fall into the “low skill” quadrants are very good candidates for automation.

Consider the following examples:

- Running a report of the previous day's sales totals is a repetitive task that requires low specialist skills. If possible, you should seek to automate this task.
- Calling a customer for a follow-up on a demo unit that was sent. This is a high-touch, high-value activity, but requires the personalization and complexity of human relationship management to execute effectively. This task may not be a good candidate for automation.

As you can imagine, using a tool such as Power Automate to reduce repetitive data manipulation tasks frees up time for people to add real value to business processes and relationships.

Next, we're going to get familiar with the terminology and interface of Power Automate and then learn how to connect to common applications to solve business problems.

## Reviewing general terminology

You've already seen a few terms, and if you're familiar with SharePoint or other collaboration tools, they may be recognizable. But just to make sure we have a solid foundation on which to build, we're going to go over some basic terminology, and then we'll start getting into specific Power Automate terminology.

### Business process

A business process is any sequence of tasks needed to accomplish the business's purpose. This may be something as simple as submitting a timecard or getting a signature on a purchase order. Business processes generally fall into three categories:

- **Primary or operating processes:** These typically result in some sort of customer value delivery, such as a customer placing an order and the business shipping a product. They also may include things such as product design and engineering.
- **Support:** These processes are necessary for the primary or operating processes to take place. For example, purchasing materials to make a product, or training an employee would generally be regarded as support processes.
- **Management:** These are processes to oversee the operating and support processes, or to improve those processes. Examples of management processes might be reviewing and making recommendations about the procurement or employee onboarding processes.

Automation can be used with processes in all of these categories. Power Automate can be used to automate some or all parts of many types of business processes.

## Workflow

A **workflow** can be thought of as the individual steps to achieve a particular business process. For example, employee onboarding may be a business process, and ordering a new computer may be a workflow task associated with completing the employee onboarding business process.

Power Automate can be used to automate some or all parts of a workflow.

## REST

**REST** is an industry acronym for **REpresentational State Transfer**. It is used to describe a method for interacting with computer systems. In a REST-based system, client devices generally send HTTP verbs (or actions) to a target system **uniform resource identifier (URI)** to input or retrieve data.



### Deep dive into REST

For more information on the history and design principles of REST architectures, refer to <https://restfulapi.net/>.

Like many technologies on the Microsoft 365 platform, Power Automate utilizes REST to enable a high volume of performant transactions.

Next, we'll look at Power Automate-specific terminology.

## Learning Power Automate terminology

As we begin working with Power Automate, it will be important to understand the core terminology that is being used. It's also important to note that terminology periodically changes, so if you started using Power Automate a few years ago, you may want to read through and make sure you're using current terminology and concepts.

The following terms are generally organized in the order that you'll encounter them in the book, which also corresponds to their increasing complexity and usage. You'll need to understand them so you can choose where to apply the correct business logic and processes. As you go through this book, if you encounter terms you don't recognize, you can refer back to this section.

## Flow

A **flow** is simply the logical grouping of conditions and tasks used to automate a process. Flows can be identified using a variety of characteristics, such as where they originate (cloud or desktop), what types of activities they perform (technical automation or business processes), when they occur (automated or scheduled), or the types of tools and technology they employ (robotic process automation or artificial intelligence).

Here are some examples of how you can describe or categorize flows:

- **Automated:** Flows that happen based on triggers or events, either cloud or desktop-based.
- **Button:** Also known as **instant** or **manual**, these occur when initiated by a user.
- **Cloud flows:** Flows created from the web interface, typically requiring no local desktop computing power or access.
- **Scheduled:** Configured to occur at specific time intervals.
- **Approval:** A process where requests are routed through an approval chain.
- **Business process:** A high-level process comprising smaller tasks and workflows.
- **Robotic Process Automation:** Also known as **RPA** or **User Interface (UI)** flows, RPA is typically used to automate non-REST apps. RPA was announced as generally available in 2020.
- **AI Builder: Artificial intelligence (AI)** technology used to help make decisions during a flow's execution. AI Builder was announced as generally available in 2020.

Each flow type has different use cases, triggers, and configuration capabilities.

## Connectors

Connectors are the components used to directly interface with both source and target systems. Connectors contain the configuration information required to interact with applications. Examples of connectors are shown in the following screenshot:

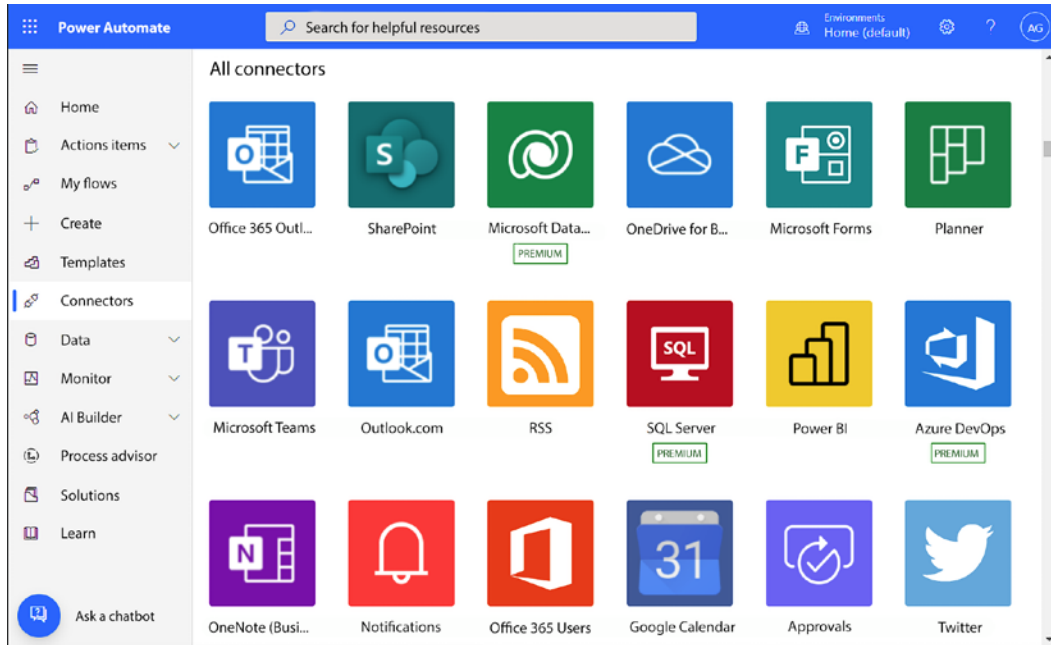


Figure 1.3: Power Automate connectors

Connectors are generally broken down into two tiers: **standard** and **premium**. Standard connectors are generally included with all Power Automate plans (such as Power Automate for Office 365), while premium connectors have additional costs associated with them (either from Microsoft or from the connector's vendor).

### Expanding Power Automate



In addition to using the included standard and premium connectors, you can also build your own connectors to interact with your applications and services. While that type of development is outside the scope of this book, you can learn more about it at <https://docs.microsoft.com/en-us/connectors/custom-connectors>.

## Triggers

Triggers are the activities that cause the flow to begin. Triggers are frequently used to categorize the type of flow being used. The core types of triggers are:

- **Automated:** An automated flow happens automatically based on a particular type of event (such as a new file being uploaded to a site or an email being received).
- **Instant:** Also known as a manually triggered flow or a button flow, instant flows are triggered on demand by a user.
- **Scheduled:** This time-based option happens on a recurring basis.

Keep in mind that the availability of automated triggers may depend on which connectors are being used – generally, your business requirement will most likely determine what type of flow and triggers you'll use.

## Steps

Steps are comprised of the individual evaluations and actions that a flow executes. They are ordered in a methodical manner. Steps frequently use output data or variables from actions performed previously. Here are some examples of steps:

- Posting a Teams channel message
- Reading an email
- Saving a file
- Performing a calculation
- Evaluating a condition
- Converting a time or date
- Entering a value in a database record

Every flow is made up of one or more steps. Each activity or process in a flow (trigger, action, or condition) is considered a step.

## Actions

Actions describe the types of activities performed by a flow (such as copying a file, posting to a Teams channel, or sending an email). From a technical perspective, a step is a group of one or more actions. Many people, however, use the terms *actions* and *steps* interchangeably.

## Conditions

Conditions are used to evaluate and select the circumstances under which actions will be performed. Conditions may operate on a number of data types, such as time or schedule constraints, values received through user input or parsing files, variables from previous actions, or other calculated values. Conditions are frequently used when the result of an evaluation needs to determine which series of actions should be performed.

## Dataverse

The **Dataverse** (previously known as the **Common Data Service** or **CDS**) is a storage mechanism similar to database tables that allows organizations to store business data. The terminology has changed several times as well, so it's important to make sure you're on the same page.

The following table shows the legacy CDS terminology as well as the more current corresponding Dataverse term:

Legacy CDS term	Current Dataverse term
Entity	Table
Field or attribute	Column
Record	Row
Option set, multi-select, or picklist	Choice
Two options	Yes/No

*Table 1.1: Terminology update*

A variety of applications, such as Dynamics 365, Power Apps, Power Automate, and Power BI can use data stored in Dataverse. Advanced Power Automate scenarios, such as working with Dynamics 365 entities, may require access to Dataverse.

## Gateways

A **gateway** (also known as a **data gateway**) is a software application installed on an on-premises computer. This application is used to facilitate access for the Power Platform services to data sources located in an on-premises environment. For example, a data gateway may be used to allow Power Automate to read items from an on-premises SharePoint Server list.

The following diagram shows the data gateway presenting on-premises data to Power Automate, Power BI, and Power Apps services:

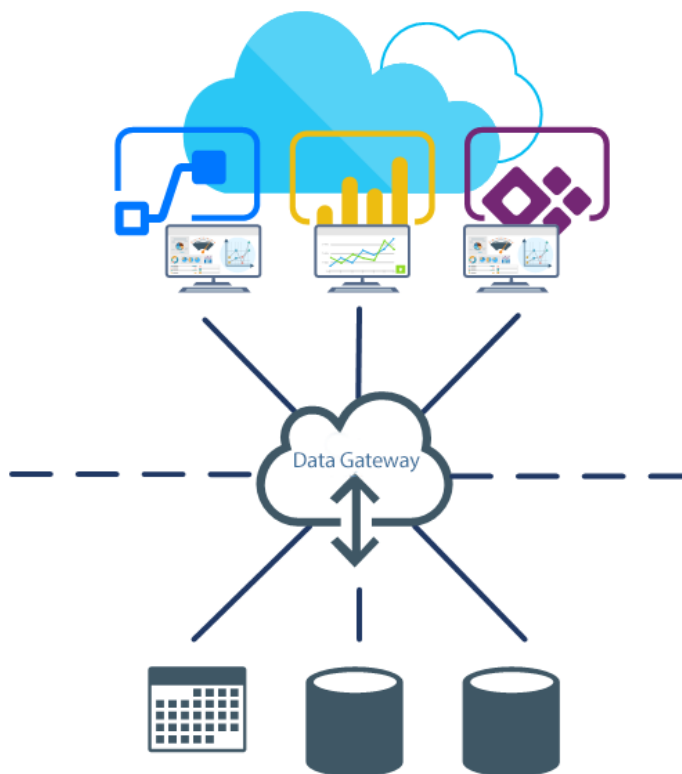


Figure 1.4: Data gateway

The data gateway, shown in *Figure 1.4*, acts as a conduit between the on-premises data sources and Power Platform services (such as Power Automate, Power BI, and Power Apps).

## Templates

A template comprises a set of connectors, triggers, and actions designed to accomplish a predefined purpose. Templates allow standardized deployments and the reuse of common components and configurations. Templates are generally published in the Power Automate template gallery (<https://us.flow.microsoft.com/en-us/templates/>).



## Packages

Similar to templates, packages allow you to save and export a flow for reuse. You can share a package either internally or externally. Packages are also useful if you develop flows in a test environment and then need a mechanism to deploy them in production.

## Branching

Branching is used to describe the concept of different series of steps or actions that can happen. You may decide to invoke branching through the use of conditions, such as in the form of an if-then construct: *if yes, do action 1; if no, do action 2*. In other instances, you may decide to use parallel branches (different sets of actions or logic that execute simultaneously) if you have long-running processes or the output of one branch isn't needed to continue the workflow of another branch.

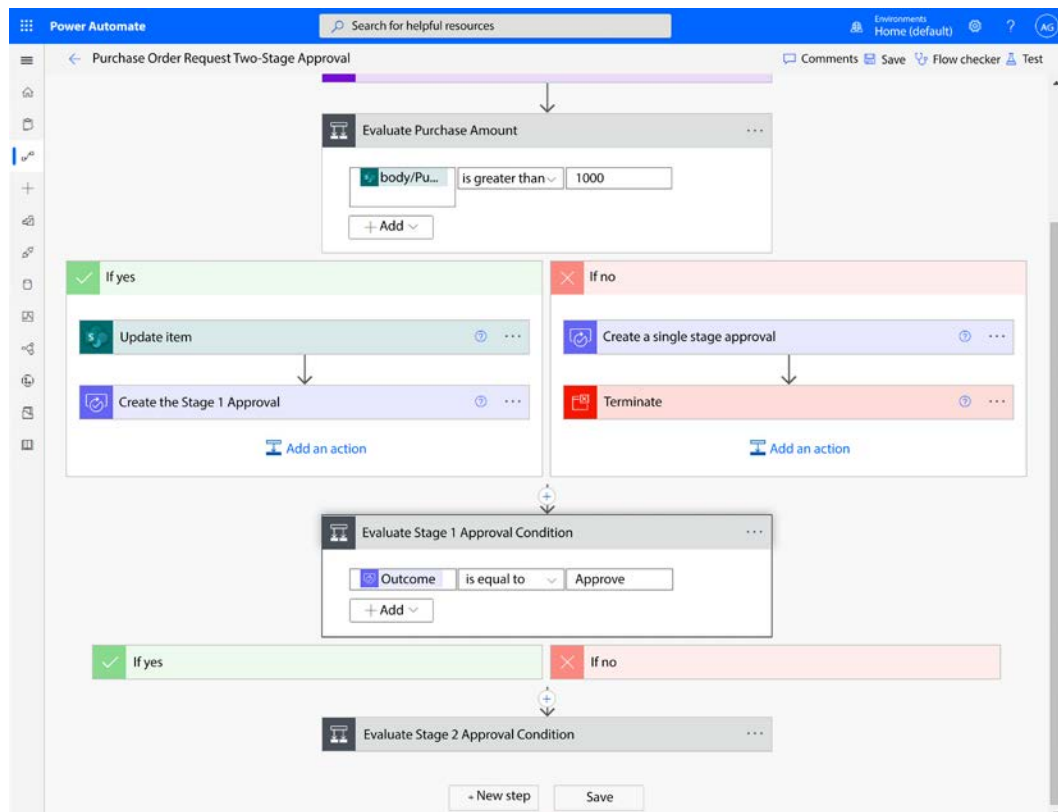


Figure 1.5: Example of branching

Understanding the terminology, both business- and Power Automate-specific, will help you as we move toward creating flows.



#### More information

While we have covered the basic Power Automate terminology you'll encounter in this book, there is even more to discover at <https://docs.microsoft.com/en-us/power-automate/glossary>.

## Getting Power Automate

Before you get started, you'll need to make sure you have licenses that support the type of flows you want to create.

As mentioned previously, Power Automate contains the concepts of *standard* and *premium* connectors. Standard connectors are available for all Power Automate plans, including the Power Automate plans that come with Office 365 (also sometimes referred to as *seeded licenses*). Premium connectors (or features such as AI Builder and RPA) may require the use of a separate standalone Power Automate license:

- Power Automate per user plan
- Power Automate per user with attended RPA plan
- Power Automate per flow plan
- Legacy Power Automate P1 or P2 plan

A per user plan is best for organizations that have broad adoption of Power Automate and wide usage. A per flow plan may be more appropriate for organizations that only have a few flows that are used by everyone in the organization. For up-to-date licensing details and guidance, see <https://docs.microsoft.com/en-us/power-platform/admin/powerapps-flow-licensing-faq>.

With that out of the way, let's dive into the platform!

## Summary

In this chapter, we discussed the high-level concepts of Power Automate, including how it relates to solving problems (such as business processes or workflows). We also covered introductory knowledge concepts such as actions, triggers, and conditions. Understanding the types of triggers and connectors will help you design flows and business processes that reduce repetitive work and produce value in your organization.

In addition to understanding the features and terminology, we covered some of the updates and changes to Microsoft Power Automate since the last revision of this book, including the introduction of Power Automate Desktop and the general availability of both RPA and AI Builder.

We also touched on the licensing necessary to take advantage of Power Automate features.

In the next chapter, we're going to begin exploring the Power Automate interface and use it to create simple flows.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 2

## Getting Started with Power Automate

In the first chapter, we introduced some of the basic concepts of Microsoft Power Automate. Now that you're familiar with some of the ideas and terminology behind Power Automate, let's take a look at navigating the interface.

As a Power Automate user, you'll have access to the web portal interface (most commonly used) as well as the desktop and mobile interfaces. If you administer a Microsoft 365 tenant, you'll also have some administrative features available.

In this chapter, we'll tackle the following:

- Logging in to Power Automate
- Creating your first flow

You'll learn the features of the interfaces, and then we'll create a basic flow to post to a Teams channel.

Let's get started!

### Logging in to Power Automate

Power Automate has four distinct experiences:

- **End user web portal:** User or creator interfaces with the purpose of designing, importing, saving, exporting, and executing flows.

- **Mobile app:** User or creator interfaces with the purpose of designing, importing, saving, exporting, and executing flows, formatted specifically for mobile devices
- **Desktop:** Most commonly used for creating **Robotic Process Automation (RPA)** flows
- **Admin:** The overall administration of the Power Automate environment for your tenant, including the number of executions or runs and data gateway configurations

Power Automate tenant administration is largely outside the scope of what we're going to be learning from a design aspect, but should you ever need to administer the Power Automate environment, you'll know where to start.

Regardless of which interface you're going to use, you'll need access to Power Automate to get the most out of this book. If you don't have access to Microsoft Power Automate within your organization, you'll want to go start a Microsoft 365 trial so you can gain access to it. To start a trial subscription, navigate to <https://www.microsoft.com/en-us/microsoft-365/try> and click the **Free trial for business link**.

It's also important to note Microsoft Power Automate is a cloud-only solution—there is no *required* on-premises server counterpart (though you may install server-side components such as the data gateway or utilize Power Automate Desktop on Windows 10 and 11 computers). Microsoft Power Automate is available in the following Microsoft 365 clouds:

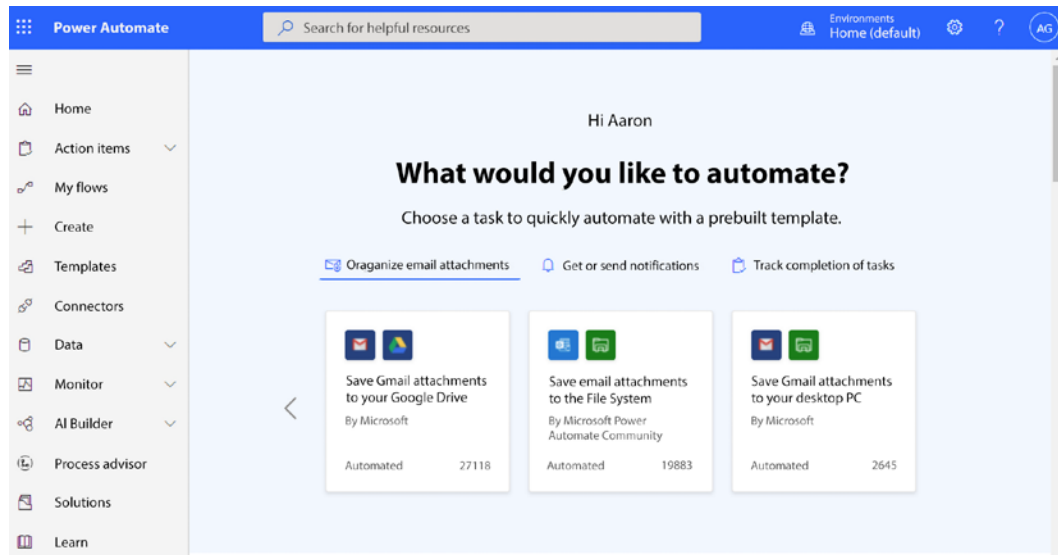
- **Worldwide/Commercial:** <https://flow.microsoft.com>
- **United States Government Community Cloud (Moderate):** <https://gov.flow.microsoft.us>
- **United States Government Community Cloud (High):** <https://high.flow.microsoft.us>

For the purposes of this book, we'll focus on the features of the worldwide commercial solution. Users of Government Community Cloud Moderate (such as state and local governments) or Government Community Cloud High (such as federal civilian employees) may have slightly different feature sets available.

## End user web portal interface

The end user web portal interface is where you'll design, manage, and execute most of your flows. To access it, you can log in to <http://flow.microsoft.com>, or log in to the Microsoft 365 end user portal (<https://portal.office.com>) and click on the **Power Automate** tile. If you're using one of the other sovereign clouds, use the links in the previous section or contact your reseller for specific links.

The web portal interface is depicted in *Figure 2.1*:



*Figure 2.1: Power Automate home page*

The left part of the page features a navigation menu that includes options to review action items (approvals and business process flows), as well as to create and manage your own flows and search for templates and connectors. The following list describes the options available:

- **Home:** The dashboard displayed upon first logging in
- **Action items:** The top-level menu item containing links to **Approvals** (approval workflows) and **Business** process flows (multi-step task-oriented flows)
- **My flows:** Flows that you have created (or that have been shared with you)
- **Create:** The interface for creating new flows
- **Templates:** Pre-canned flows comprising connectors that only need to be customized
- **Connectors:** List of connectors available in Power Automate
- **Data:** A top-level menu item containing links to the following:
  - **Tables** (formerly **Entities**): Objects in Dataverse (formerly Common Data Service)
  - **Connections:** Authentication configurations for connectors

- **Custom connectors:** Custom interfaces to REST API applications
- **Gateways:** Basic information about existing data gateways to connect to on-premises data sources
- **Monitor:** A top-level menu item containing links to the following:
  - **Cloud flow activity:** Review recent cloud flow runs or executions
  - **Desktop flow runs:** Review recent desktop flow runs
  - **Machines:** Real-time status for machines running desktop flows
- **AI Builder:** A top-level menu item containing links to the following:
  - **Explore:** Samples, examples, and pre-built models for learning and using in your own flows
  - **Models:** AI models that you've built or have been shared with you for creating AI-based flows
  - **Document automation:** Tools for using both AI and RPA to extract structured and unstructured data from documents
- **Process advisor:** Tools for analyzing business processes and detecting automation opportunities to be optimized
- **Solutions:** Containers for managing the transporting and life cycle of Power Platform applications
- **Learn:** A link to the landing page for Power Automate in the Microsoft documentation (<https://docs.microsoft.com/en-us/power-automate/>)

In this book, we'll explore some of the components to use when building your own flows. You may want to spend some time looking through the connectors (<https://flow.microsoft.com/en-us/connectors/>) and templates (<https://flow.microsoft.com/en-us/templates/>) to start getting ideas of what types of applications and services you can integrate with Power Automate. You'll want to focus on connectors and templates that help you integrate apps that your organization uses on a regular basis.

## Mobile app interface

The Power Automate mobile app, available for both the iOS (<https://apps.apple.com/us/app/microsoft-flow/id1094928825>) and Android (<https://play.google.com/store/apps/details?id=com.microsoft.flow>) platforms, offers a similar design aesthetic and experience, as shown in the following screenshot:

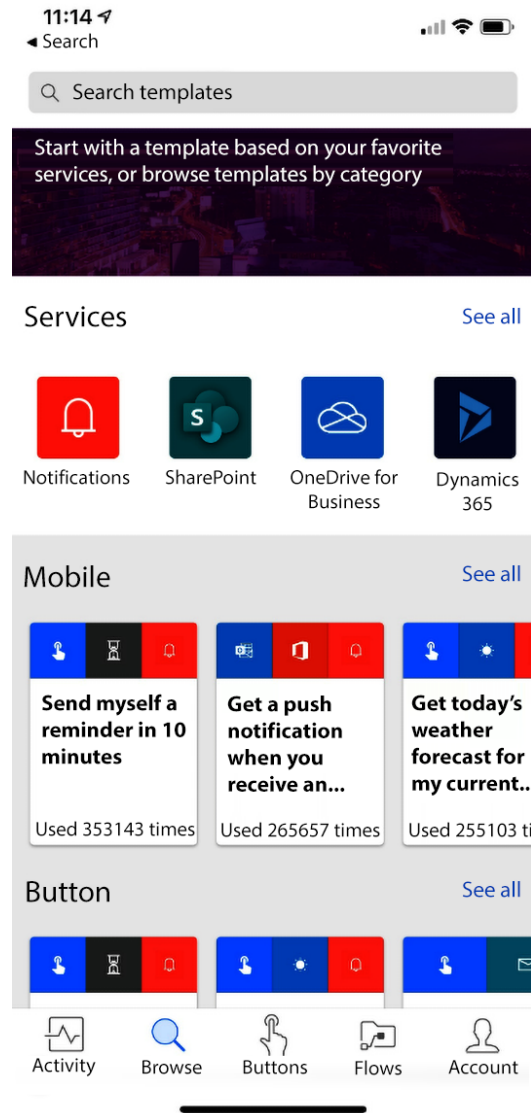


Figure 2.2: Power Automate for iOS

The following options are available on the home screen:

- **Activity:** Shows a list of recently completed flows and approval activities
- **Browse:** Search connectors and templates to begin creating a new flow
- **Buttons:** Allows you to create button (or instant) flows to be triggered from your mobile device



- **Flows:** Lists your own and your team's (that is, shared) flows
- **Account:** Your Office 365 tenant account sign-in information

The mobile experience focuses on surfacing flow components specifically tailored to mobile devices. The majority of this book will focus on the end user web portal experience, but there will be a few opportunities to use the mobile app.

## Power Automate Desktop interface

As mentioned in *Chapter 1, Introducing Microsoft Power Automate*, Power Automate Desktop is a new way to interact with flows. You can only create robotic process automation flows from Power Automate Desktop.

The Power Automate Desktop application itself is free to download for Windows 10 and is included with Windows 11. However, in order to use advanced features (such as RPA flows), you may need to acquire additional licensing.

Figure 2.3 shows the Power Automate Desktop application on Windows 10:

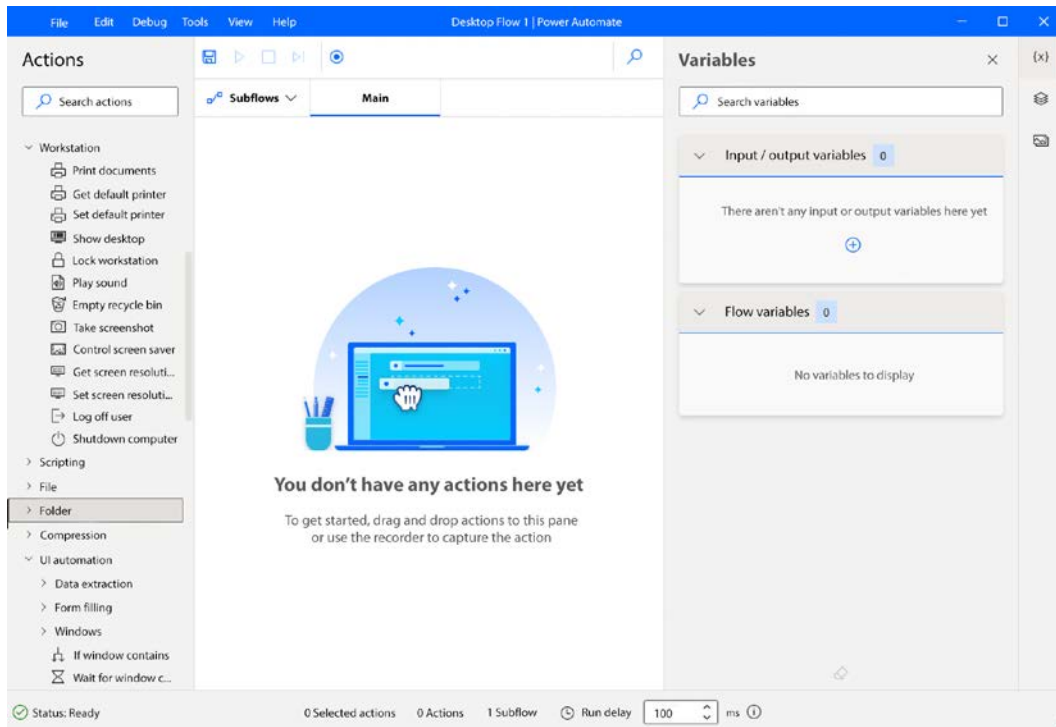


Figure 2.3: Power Automate Desktop on Windows 10

Power Automate Desktop allows you to interrogate system resources and interact with files, browser instances, applications such as Outlook and Excel, command prompts, and even printers.

## Admin interface

Previously, administrators needed to use both the Power Automate admin center as well as the Power Platform admin center to manage their environments. In 2021, all the functions were combined into the Power Platform admin center. You'll have view-only access to this interface unless you're a global admin of a Microsoft 365 tenant or have been granted Power Platform admin rights.

The Power Automate admin center (previously <https://admin.flow.microsoft.com>, which now redirects to <https://admin.powerplatform.microsoft.com>) allows you to keep track of runs executed tenant-wide, view license information, view overall environment data, and configure **Data Loss Prevention (DLP)** policies. In addition, the Power Platform admin center allows you to administer features across the entire Power Platform, including Power Automate, but also to Dynamics 365, Power BI, and Power Apps, as shown in *Figure 2.4*:

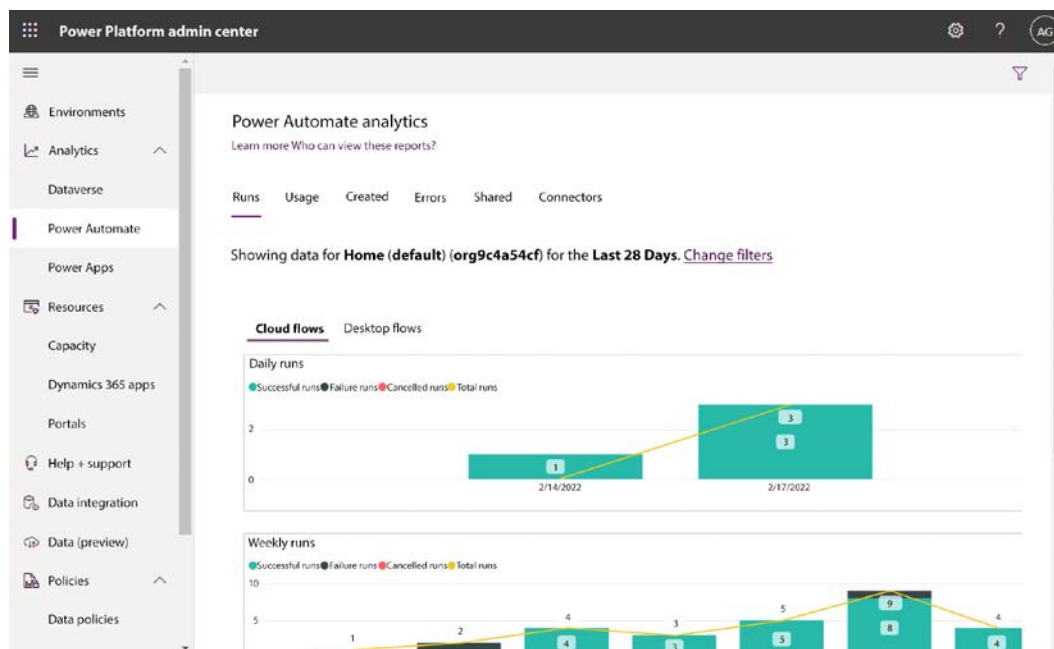


Figure 2.4: Power Platform admin center

While the admin interface will not be used extensively in this book, you should understand that it exists and that administrators of the overall Microsoft 365 environment can modify configurations and view statistical data across the organization.

Now that you've seen the various interfaces for Power Automate, let's start creating!

## Creating your first flow

The best way to see Power Automate in action is to start creating a flow. In this example, we're going to create a flow that monitors Twitter for a certain hashtag and then posts a notification to a Teams channel. Such a flow might be useful if you're trying to gauge or capture customer sentiment for a product or service, track trending public health topics related to certain keywords, monitor engagement activity, or other topic-based alerts on a social media platform.

## Understanding the flow components

This particular flow is going to rely upon a few components:

- An identity for Twitter (username and password)
- A trigger that monitors Twitter for certain words or phrases
- An identity for Office 365 (username and password)
- A Microsoft Teams team
- An action that posts to Microsoft Teams

To complete this flow, you'll need your existing Microsoft 365 identity (or the identity you're using in a test environment), as well as an identity for Twitter (<https://www.twitter.com>). You'll also need access to a Microsoft Teams team where messages will be posted.



### About Microsoft Teams

If you're not familiar with **Microsoft Teams**, it's a modern collaboration software platform that brings several Microsoft technologies under a single, unified user interface. You can learn how to create teams here: <https://docs.microsoft.com/en-us/microsoftteams/get-started-with-teams-create-your-first-teams-and-channels>. You can also learn more about Microsoft Teams in another Packt book, *Expert Microsoft Teams Solutions* (ISBN: 978-1801075558).

Once you have the required identities configured, we'll start creating the flow!

## Creating and executing the flow

Once we have the prerequisite components for the flow (that is, the aforementioned identities and a team), we can begin the configuration. To build this sample flow, follow these steps:

1. Log in to the Power Automate web interface (<https://flow.microsoft.com>).
2. On the left-hand side of the page, click **Create**.
3. On the **Manage your flows** page, under **Start from blank**, click **Automated cloud flow**:

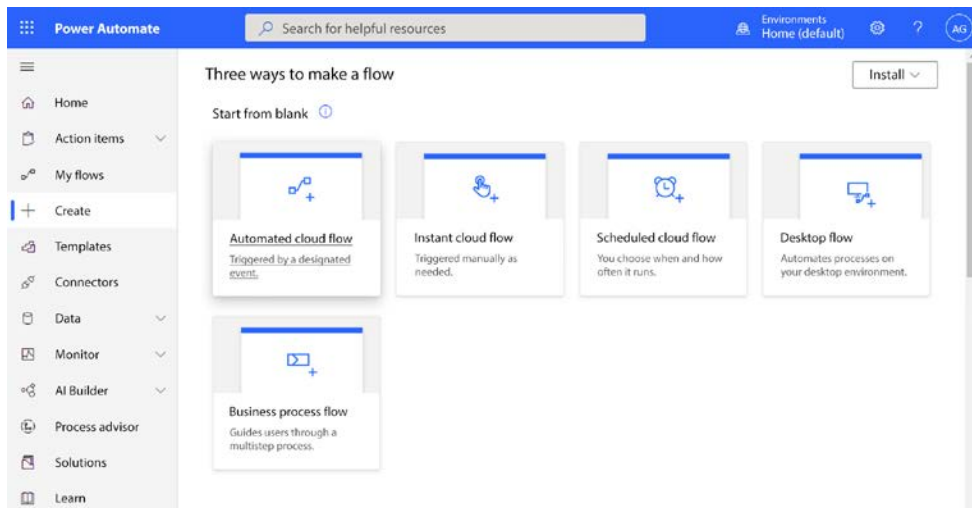


Figure 2.5: Creating an automated cloud flow

4. Enter a **Flow name**.

5. In the **Choose your flow's trigger** box, start typing Twitter to filter triggers related to Twitter. Select **When a new tweet is posted** and click **Create**:

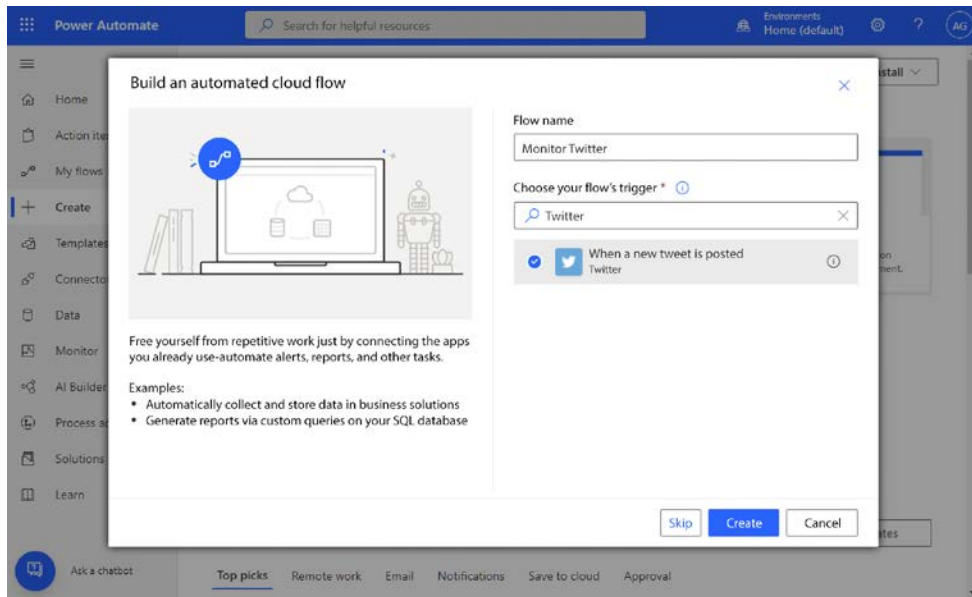


Figure 2.6: Creating a new cloud flow

6. Next, click **Sign in** to provide the necessary Twitter credentials to access the Twitter API. You can use the **Use default shared application** authentication type for this basic flow:

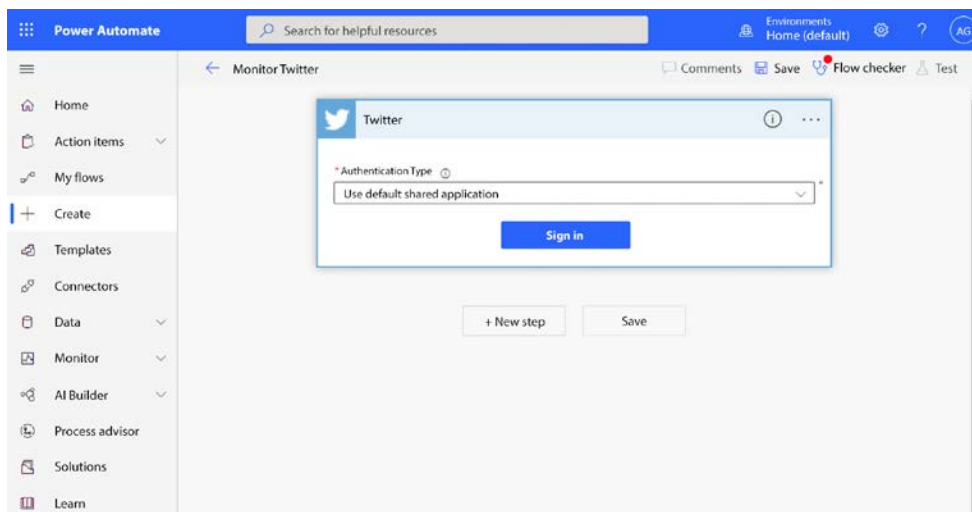
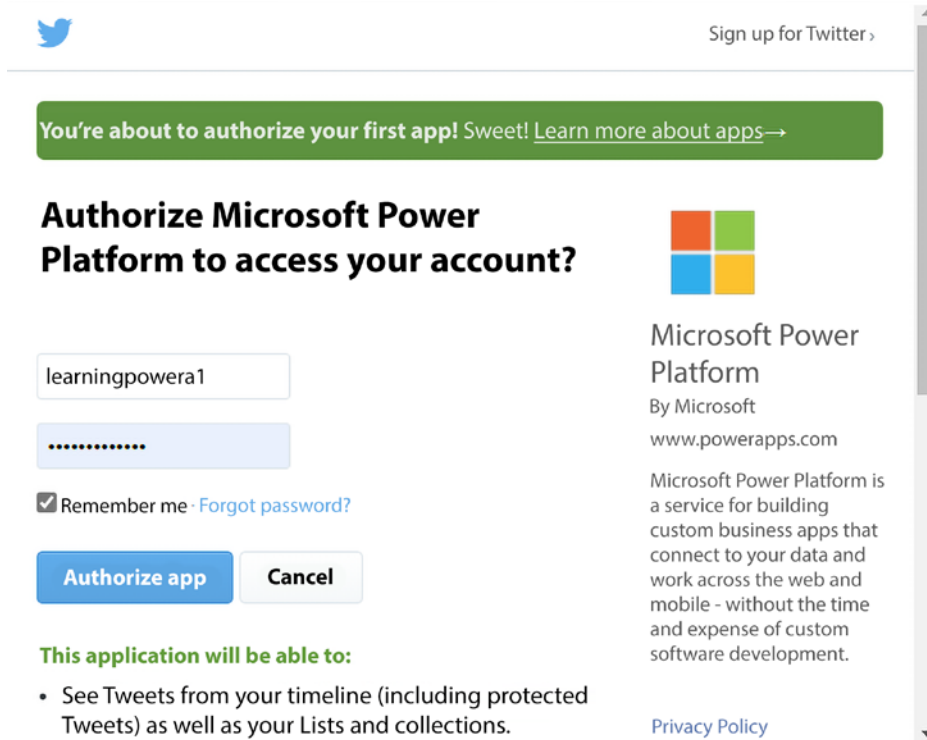


Figure 2.7: Entering Twitter credentials

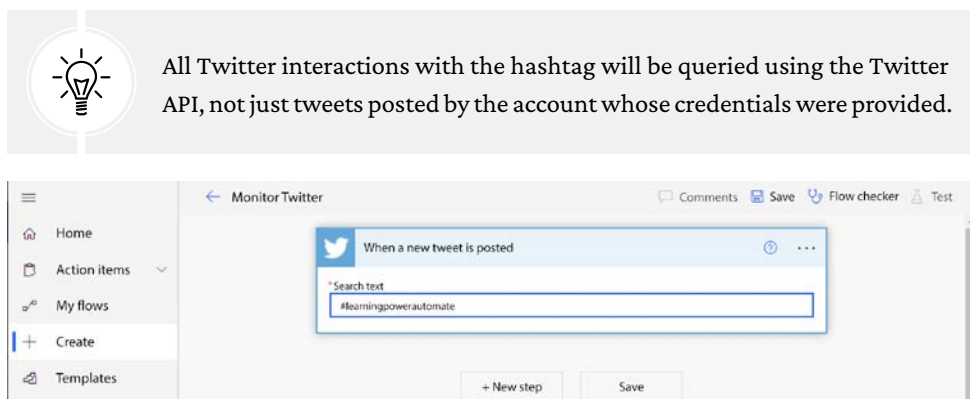
7. Enter the credentials of a Twitter account and then select **Authorize app**:



The screenshot shows the Twitter authorization interface. At the top, there's a Twitter logo and a "Sign up for Twitter" link. Below that, a green banner says "You're about to authorize your first app! Sweet! [Learn more about apps](#)→". The main heading is "Authorize Microsoft Power Platform to access your account?". To the right is the Microsoft Power Platform logo and text: "Microsoft Power Platform", "By Microsoft", "www.powerapps.com", and a description: "Microsoft Power Platform is a service for building custom business apps that connect to your data and work across the web and mobile - without the time and expense of custom software development." Below the heading, there's a text input field containing "learningpowera1", a password field with dots, and a checkbox labeled "Remember me" with a link "Forgot password?". There are two buttons: "Authorize app" (blue) and "Cancel" (gray). At the bottom, it says "This application will be able to:" followed by a list: "• See Tweets from your timeline (including protected Tweets) as well as your Lists and collections." and a "Privacy Policy" link.

Figure 2.8: Authorizing the Twitter application

8. In the **Search** text box, enter the text that you want to search for. In this case, the flow will be configured to monitor new instances of #learningpowerautomate:



The screenshot shows the Microsoft Flow interface. On the left is a sidebar with navigation options: Home, Action items, My flows, Create, and Templates. The main area is titled "Monitor Twitter" and has tabs for "Comments", "Save", "Flow checker", and "Test". A configuration window for the "When a new tweet is posted" trigger is open. It has a "Search text" input field containing "#learningpowerautomate". At the bottom of the configuration window are buttons for "+ New step" and "Save". Above the configuration window, there's a lightbulb icon and a text box stating: "All Twitter interactions with the hashtag will be queried using the Twitter API, not just tweets posted by the account whose credentials were provided."

Figure 2.9: Entering search text

9. Click **New step**.
10. Under **Actions**, search for the Microsoft Teams **Post message in a chat or channel** action and select it:

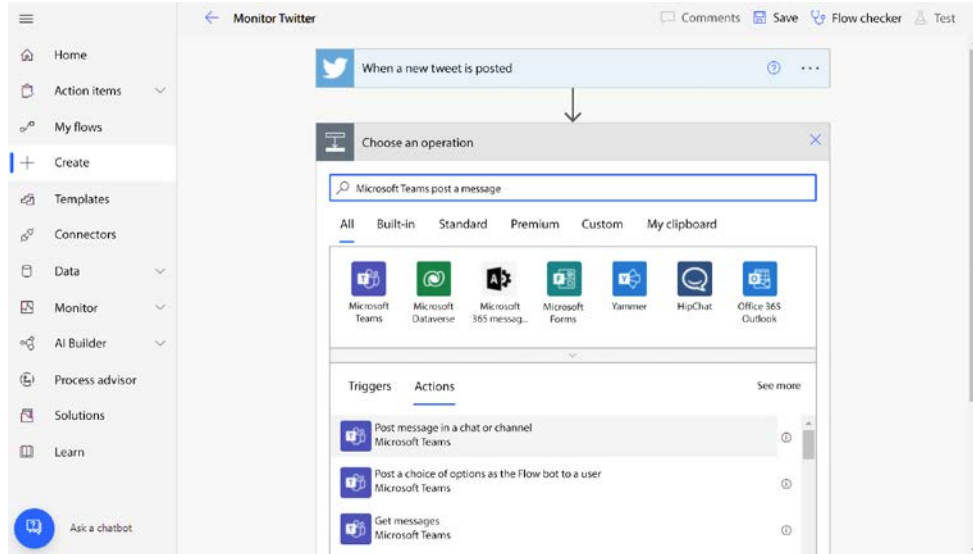


Figure 2.10: Selecting the Post message action

11. Under **Post as**, select **Flow bot**.
12. Under **Post in**, select **Channel**. This will allow you to select a team and a channel to post the message. Alternatively, you could also select **Chat with Flow bot** (some of the screens will be different). Select a **Team** and **Channel** where you have access to post messages.

13. Populate the **Message** field. In this example, the dynamic content values **Tweeted by** and **Tweet text** have been selected to surface data provided by Twitter:

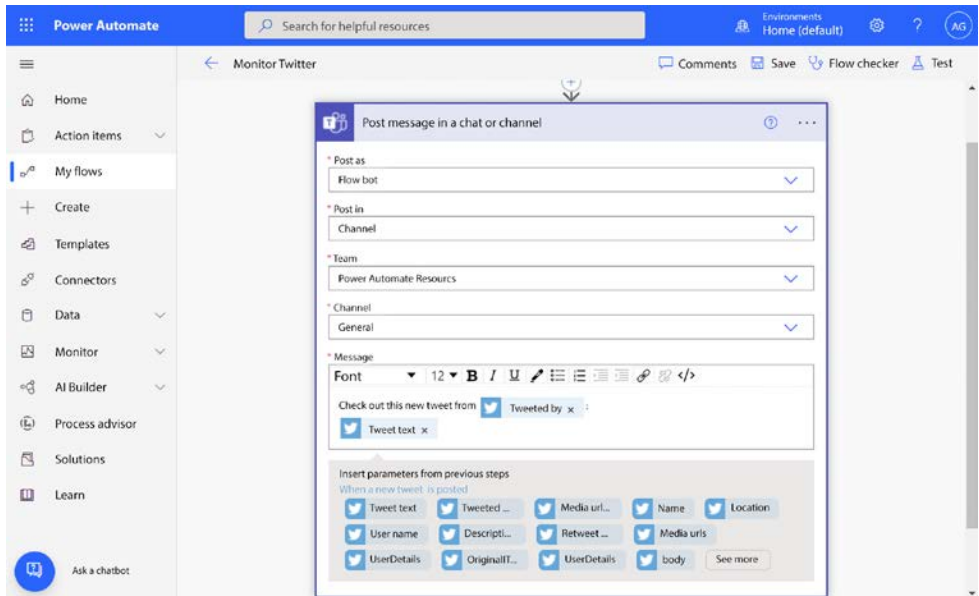


Figure 2.11: Filling out the Post message action

### About Dynamic Content



Many actions will give you access to *dynamic content* (data that is automatically populated, calculated, or retrieved) based on the context and parameters of the flow's connected data sources and actions. Dynamic content values are automatically generated variables with content based on the result of previously completed actions or triggers.



14. Scroll to the bottom of the page and select **Save**.

Now that the flow has been created and saved, you can test it by generating content and then checking to see whether the content gets posted to the Microsoft Teams channel as you'd expect:

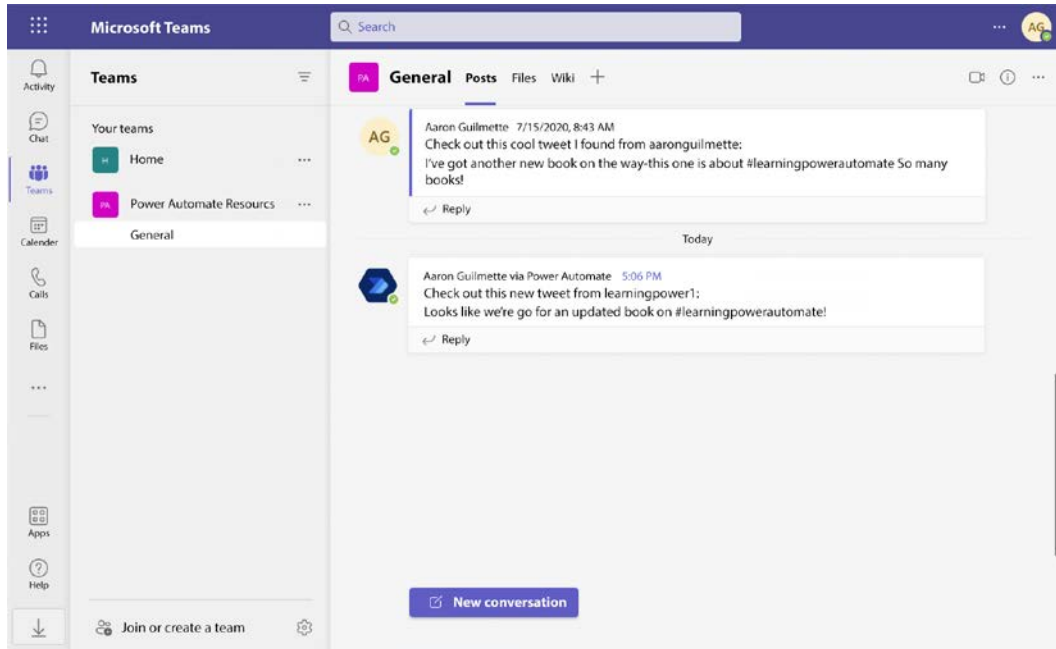


Figure 2.12: Checking Microsoft Teams after tweeting

If it doesn't look the way you expect, you can try adding different dynamic content variables or investigating the data retrieved during the run. We'll examine some of the troubleshooting processes in *Chapter 19, Monitoring and Troubleshooting Flows*.

## Examining the flow

At any time, you can use a feature called **Peek code** that allows you to look at the underlying **JavaScript Object Notation (JSON)** code that controls how a step functions.

To access the JSON view for a step, click the **ellipsis (...)** for the step and then select **Peek code**:

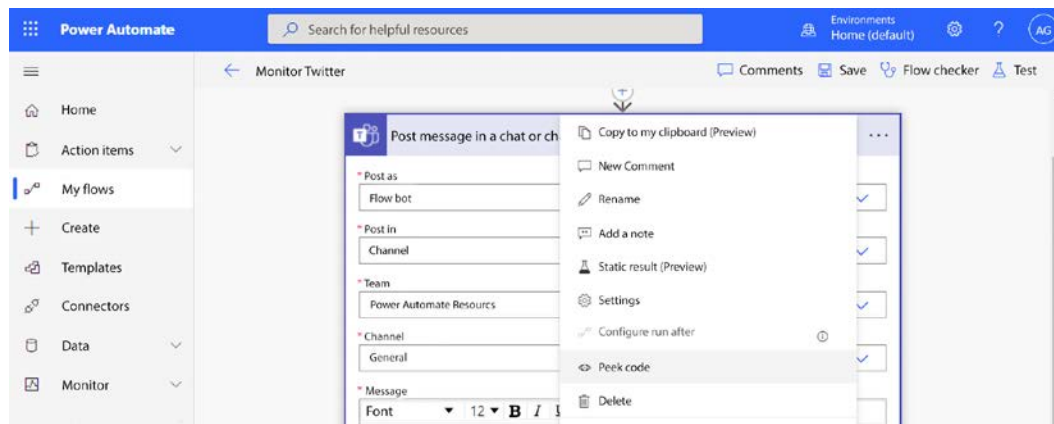


Figure 2.13: Launching Peek code

You can then navigate the detailed code for the step.

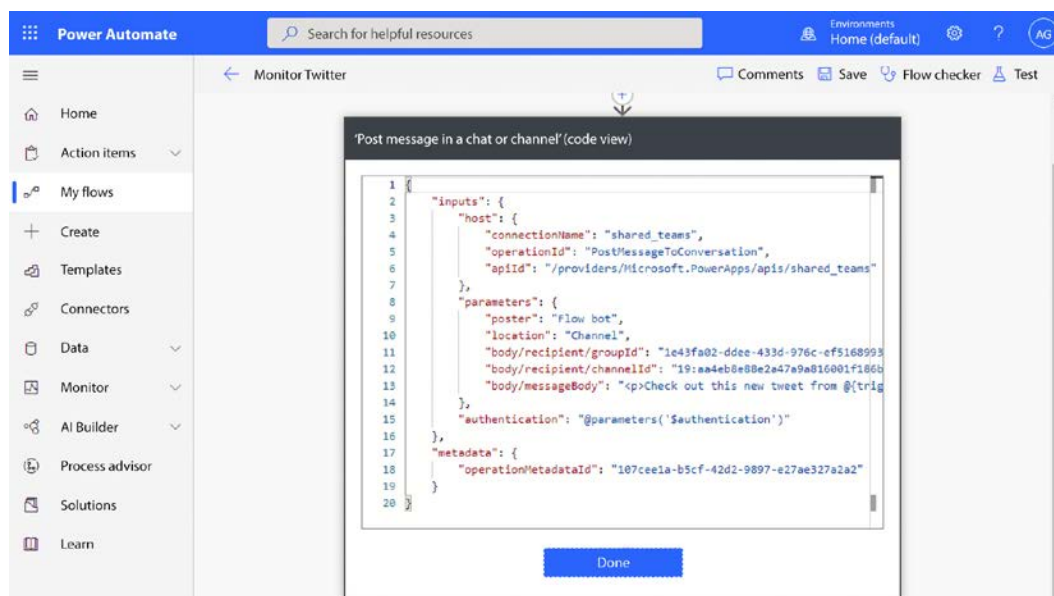


Figure 2.14: Viewing the JSON for a step



You can't directly edit the JSON code in Power Automate's user interface, but you can export it and edit it in an editor such as Visual Studio.

This flow built on your basic knowledge of flow components by adding dynamic content. We'll use a lot more dynamic content in later chapters. As you can see, being able to use dynamic content placeholders and references will increase the richness of the flows you create.

When finished, you can click **Done** to leave the JSON code view.

## Summary

In this chapter, you learned about the various Microsoft Power Automate interfaces (web, mobile, desktop, and admin) and some of the features available in each of them. Using the knowledge of Power Automate and the workflow concepts introduced in *Chapter 1, Introducing Microsoft Power Automate*, we built a simple flow to generate a Teams channel message based on Twitter keywords.

Our simple flow illustrated the power of automation to help organizations stay connected and generate actionable information in a relatable manner.

In the next chapter, we're going to begin using Power Automate to perform common email tasks, such as filtering messages and working with attachments.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 3

## Working with Email

Over the years, many business processes have been developed that use email as a storage, tracking, or processing mechanism. These processes have relied upon a combination of manual activities, third-party plugins, **Component Object Model (COM)** add-ins, and **Visual Basic for Applications (VBA)** scripting. As each version of email software changes, the technology and methods have had to change.

Instead of using discrete or custom code modules, Power Automate uses open REST-based APIs to interact with data for web services. As you learned in *Chapter 1, Introducing Microsoft Power Automate*, connectors are used to attach to data sources and endpoints (either to receive, retrieve, store, or send data).

Connectors are structured configuration files (typically formatted as **JavaScript Object Notation** or **JSON**) that define how Power Automate will interact with another service for automation tasks.

So what makes Power Automate different from previous automation technologies?

A Power Automate-based solution's strength lies in the cloud, as it doesn't rely on any desktop or local computer components to be installed or running for most automation tasks (though elements can be installed to access on-premises datasets or manipulate legacy applications). Power Automate isn't dependent on particular versions of a desktop client and can be built in a codeless (or near codeless) manner using an extensible, componentized architecture.

This chapter will focus on using Power Automate to manage email:

- Learning about email connectors and actions
- Working with email

By the end of the chapter, you should be able to connect Power Automate to an email account and perform common tasks such as saving attachments and sending messages.

## Learning about email connectors and actions

One of the most common scenarios for end user automation is email. Microsoft Power Automate can connect natively to different types of email systems, such as Office 365, Outlook.com, and Gmail.

Additionally, there are third-party plugins such as *Mailparser* that further enhance the ability of Microsoft Power Automate to process messages.

When processing a mailbox, there are several types of actions available, depending on the data source. Popular actions include the following:

- Creating or deleting a calendar event
- Creating or deleting a contact
- Flagging an email (such as the Importance flag)
- Forwarding and replying to an email or sending a new one
- Marking an email as read
- Downloading an attachment

As you saw in the previous chapter, you are able to access *dynamic content* – properties, metadata, and content related to a unique item. When manipulating mailbox content, you have many types of objects available to you as dynamic content parameters.

For example, when retrieving messages using the **Get Emails (V3)** action ([https://docs.microsoft.com/en-us/connectors/office365/#get-emails-\(v3\)](https://docs.microsoft.com/en-us/connectors/office365/#get-emails-(v3))), you can work with these common parameters of a message:

- **Folder** (the folder containing a message)
- **To** (the recipients of a message)
- **CC** (the CC recipients of a message)
- **From** (the sender of a message)
- **Importance** (message importance)
- **Subject** (message subject)
- **Search Query** (message filtering)

By working with those parameters or fields in the form of dynamic content, you can create flows with customized or personalized information, as well as evaluate the parameters to ensure flow only executes when certain conditional constraints are met.



For a complete list of the hundreds of connectors and actions available, see <https://docs.microsoft.com/en-us/connectors/>.

In the next section, we're going to start exploring some of the possibilities of the Office 365 email connector (though many of the functions and concepts will be applicable when working with Outlook or Gmail connectors).

## Working with email

The challenge of automating email has been around almost as long as email itself. As we mentioned at the beginning of the chapter, many solutions have been developed to address this over the years using technologies such as VBA scripting, Outlook rules, and COM add-ins. These solutions have required a mix of programming skills to develop and a dependence on a desktop computer always running to be able to execute the commands.

With Power Automate and Microsoft 365, Outlook.com, or Gmail services, the necessary components are *always on*, so you don't need to worry about power outages, software updates, or network connectivity issues disrupting a business process. In this section, we're going to look at options for handling incoming emails, processing attachments, and sending messages.

## Reading email

Many organizations use the receiving of an email (typically, mail sent to a shared mailbox) as an informal trigger to start a business process. This could be a sales order, a service ticket request, or a request for some sort of information. In this example, we're going to look at some of the filtering and selection options available when processing an incoming message.

To work with this flow, we're going to need an Exchange Online mailbox and a Microsoft Power Automate subscription. If you signed up for a Microsoft 365 trial, you have everything you need!

For this scenario, we're just going to go through a few steps to see how we can use filtering parameters to identify messages that could be related to an expense report. Follow these steps:

1. The easiest way to start is to log into the Power Automate web portal (<https://flow.microsoft.com>), click **Create**, and then select **Automated cloud flow**.

2. Enter a value for **Flow name**, and then search the triggers for **When a new email arrives (V3)**:

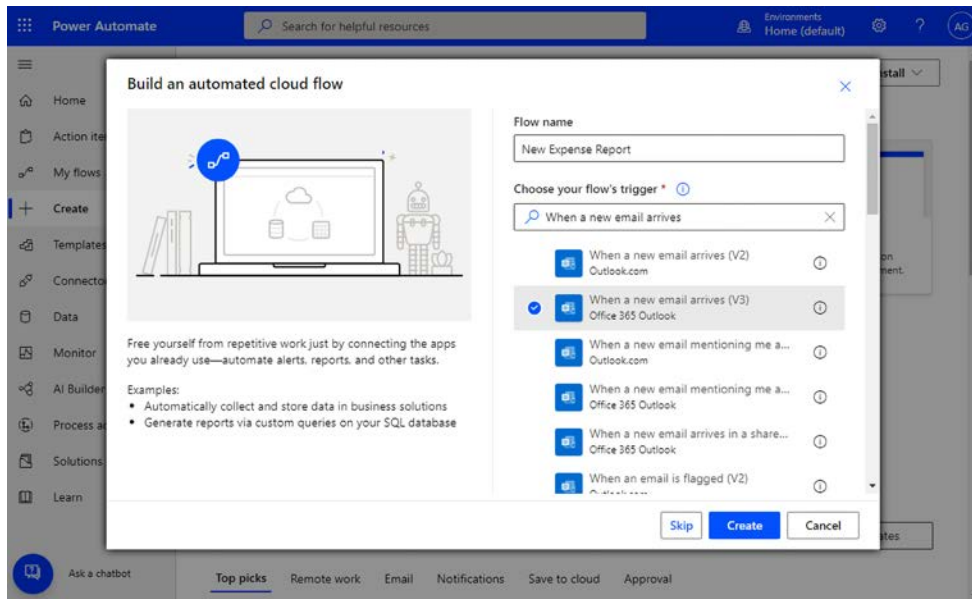


Figure 3.1: Creating a new automated cloud flow



There are also triggers for **When a new email arrives in a shared mailbox (V2)** and **When a new email arrives to a group**. Any of them can be used, but you'll need to ensure that the account you use as a credential has access to the shared mailbox or is a member of the group being monitored. When we create a complete flow later on in this chapter, we'll use the shared mailbox option.

3. Click **Create**.
4. By default, the flow is going to connect to the Outlook mailbox for the currently logged-in account. If you need to monitor a different mailbox, select the ellipsis (...) icon on the **When a new email arrives (V3)** step, and then click **Add new connection** to provide the necessary mailbox name and credentials:

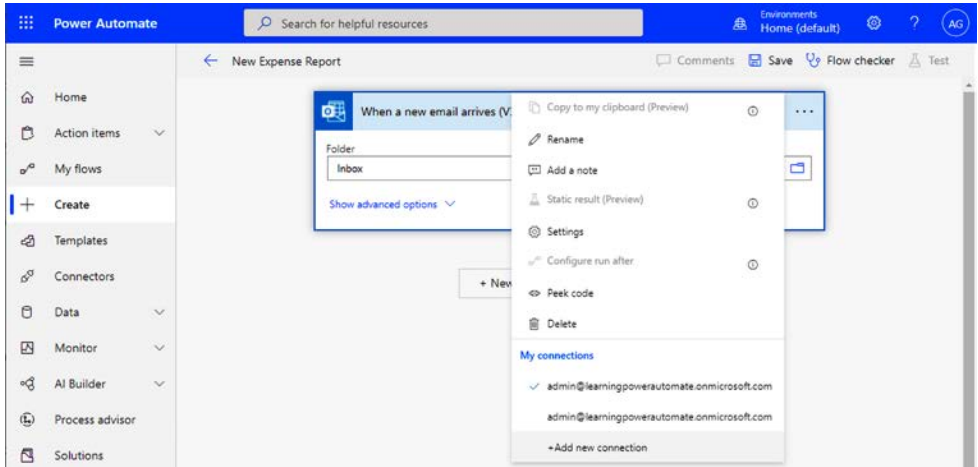


Figure 3.2: Adding a credential to connect to an additional mailbox

5. Click **Show advanced options** to expand the dropdown.
6. In the **Subject Filter** box, enter a text value such as **expense report**. This will ensure that the flow only performs actions on specific messages:

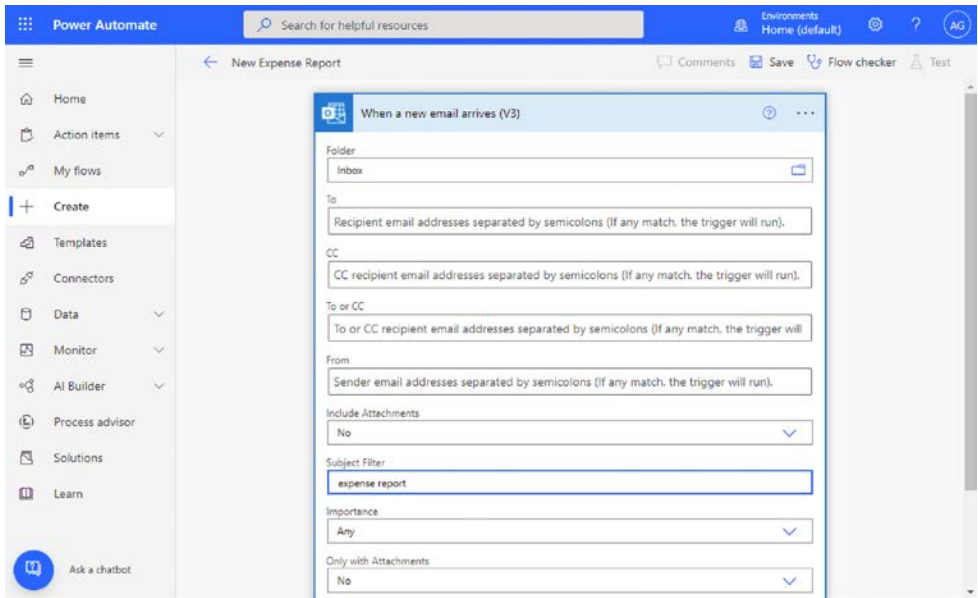


Figure 3.3: Adding a subject filter



7. Click the back arrow next to the title of the flow to exit. Since we're just using this to learn about some of the options available, confirm that you don't want to save it.

As you saw in the previous example, you can very quickly narrow down the messages that the flow would act upon. The **When a new email arrives (V3)** trigger has several configurable options, including which folder to check for new mail, sender or recipient selections, importance or priority flags, and attachment settings.

#### Similar, but not matching



When you were creating the flow, you probably also noticed other email-based triggers that looked similar: **When a new email mentioning me arrives (V2)** and **When a new email mentioning me arrives (V3)**.

The triggers for **When a new email mentioning me arrives** use a special Microsoft Graph API call to check for messages with an @mention (pronounced *at mention*) for your identity. They're good for finding specifically crafted messages (like messages where you may be assigned a task). Using an @mention for a recipient adds them to the **To:** line.

We'll take a look at some additional conditional filtering concepts in *Chapter 8, Working with Conditions*.

## Handling attachments

In addition to being able to process text in an email, many use cases require the ability to save and manipulate attachments. This type of request is frequently seen in mailboxes used to receive things such as purchase orders or expense reports. You can use Power Automate to save those items to OneDrive or a SharePoint document library, or further act upon them with other connectors, actions, or manual intervention.

Sometimes, you may even be faced with situations where you need to process multiple attachments to a single email message. Fortunately, Power Automate has some built-in intelligence and capability for this as well.

In this next example, we're going to put together a sample flow that does the following:

- Monitors a shared mailbox for messages that contain the words *expense report*
- Saves the attachment to a SharePoint Library called *ExpenseReports*
- Marks the email as *Read*
- Marks the email as *Complete*

To prepare for this example, you'll need the following:

- A SharePoint site with a document library in which you can save files. If you're unsure about how to create a site or a library, you can follow the steps at <https://support.microsoft.com/en-us/office/create-a-document-library-in-sharepoint-306728fe-0325-4b28-b60d-f902e1d75939> to create one or you can use an existing document library connected to a Microsoft team.
- A shared mailbox to which you've been granted management permissions. If you're unsure about how to create a shared mailbox, you can follow the steps at <https://docs.microsoft.com/en-us/exchange/collaboration-exo/shared-mailboxes> or just use a regular user mailbox. You may want to refer to the previous example's steps for working with a regular mailbox.

If you signed up for a new Microsoft 365 trial subscription to follow along with this book, you'll have access to everything you need. With those requirements met, you can begin creating the flow.

## Creating the flow

In order to complete this example, follow these steps:

1. Log into the Power Automate web portal (<https://flow.microsoft.com>), click **Create**, and then select **Automated cloud flow**.

2. Enter a value for **Flow name**, and then search the triggers for **When a new email arrives in a shared mailbox (V2)**:

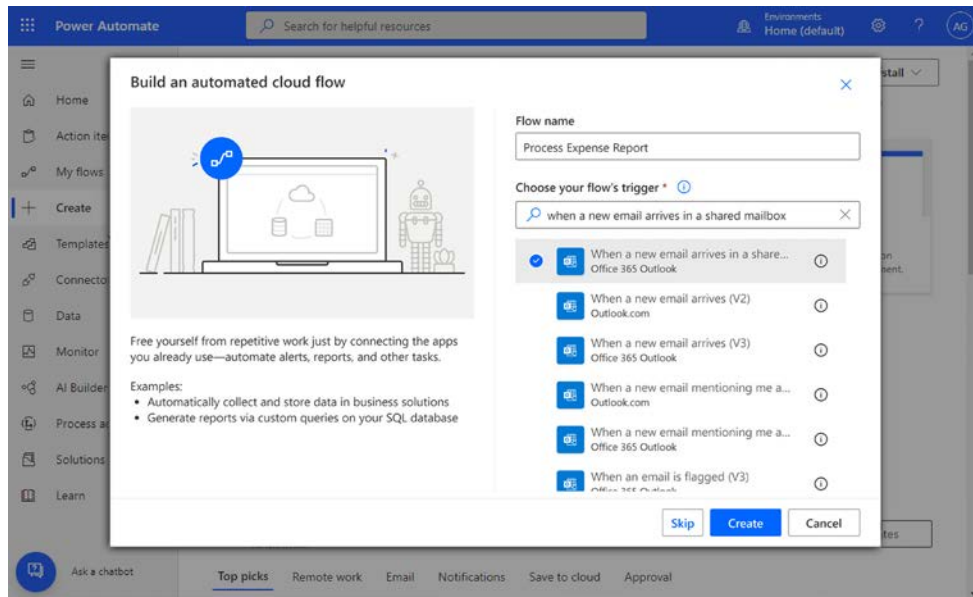


Figure 3.4: Creating the Expense Report flow

3. In the **Mailbox Address** box, enter the address of the shared mailbox you wish to monitor, as shown in Figure 3.5:

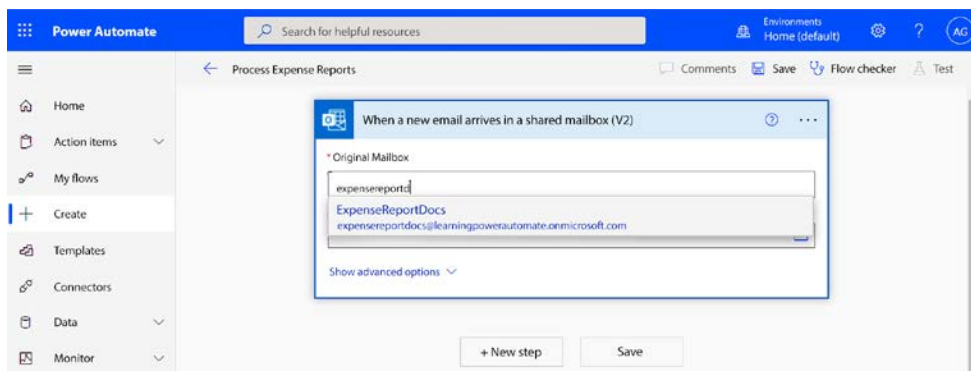


Figure 3.5: Adding a shared mailbox

4. Click **Show advanced options** to expand the options.
5. In the **Only with Attachments** box, select **Yes**. In the **Include Attachments** box, select **Yes**. The **Only with Attachments** option configures the step to only work with messages that have attachments, while the **Include Attachments** choice will make the attachment content available later in the flow:

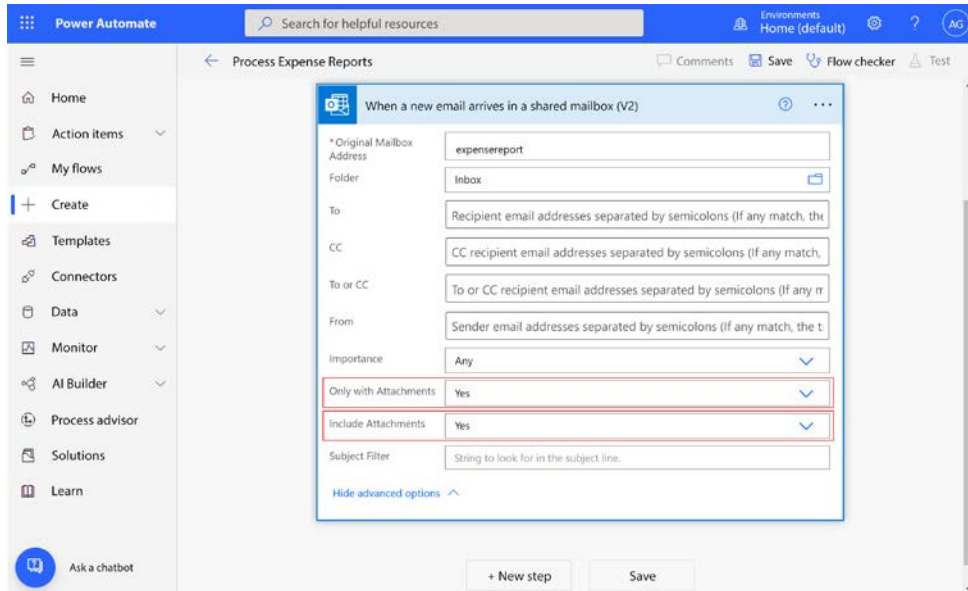


Figure 3.6: Setting attachment filter options

6. In the **Subject Filter** box, enter **expense report**. This will place a further constraint on the rule to only work if all three of the conditions are met.
7. Click **New step**.

8. Enter **sharePoint create file** in the **Choose an operation** box and select the **Create file** action for SharePoint:

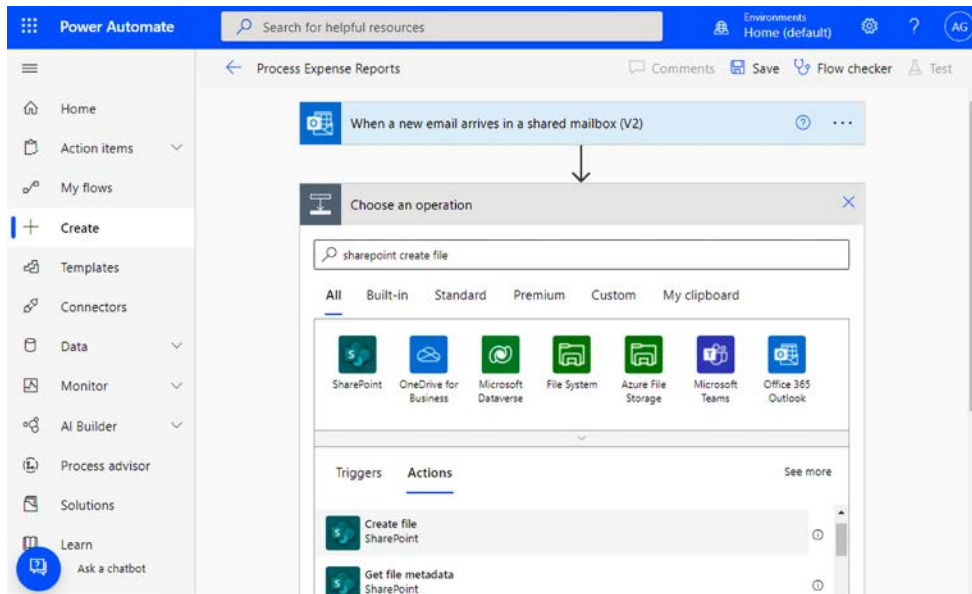
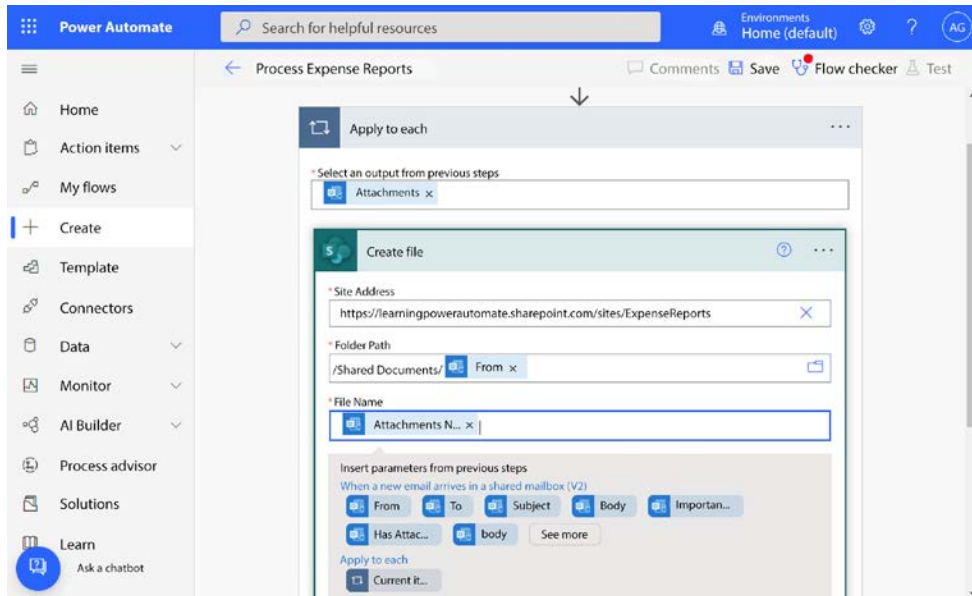


Figure 3.7: Selecting the SharePoint Create file action

9. In the **Site Address** box, add the URL for the SharePoint site that will store the document.
10. In the **Folder Path** box, click the folder icon to browse for a document library folder such as **Shared Documents**. In this example, we added a trailing forward-slash (/) and then used the dynamic content variable **From** to create a subfolder for each email sender.

11. In the **File Name** box, select the dynamic content object **Attachments Name**. As soon as you select **Attachments Name**, Power Automate will automatically wrap your step in the **Apply to each** function, as it is aware that an email may have more than one attachment:



*Figure 3.8: Selecting Attachments Name causes the Apply to each function to be invoked*

12. In the **File Content** box, select the dynamic content object **Attachments Contents**.
13. Outside the **Apply to each** function, click **New step**.

14. Enter the search text **Mark as read** and select the **Mark as read or unread (V3)** action:

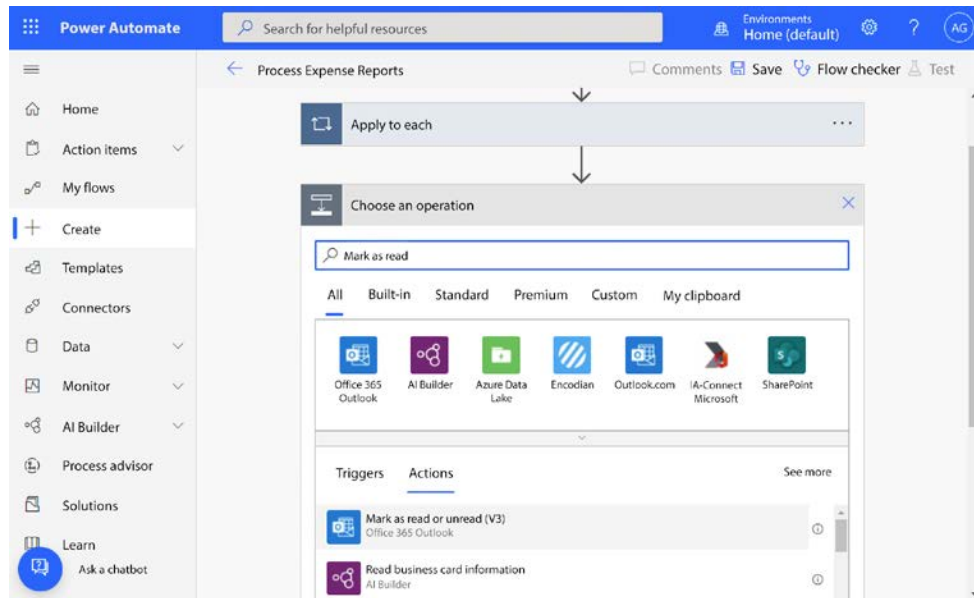


Figure 3.9: Adding a new action

15. Select the **Message Id** box, and then browse to the dynamic content object **Message Id**. This will select the current message being processed:

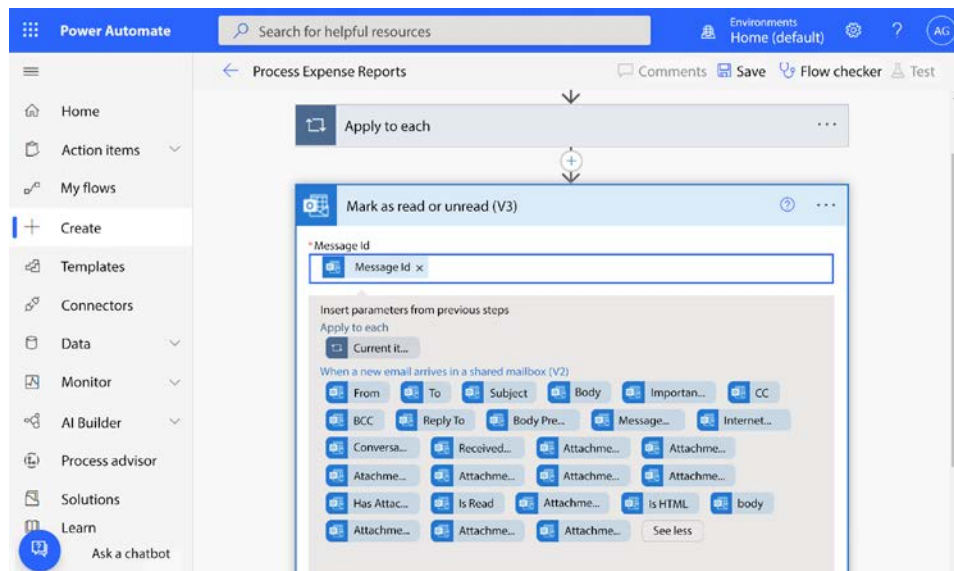


Figure 3.10: Selecting the Message Id dynamic content token

16. In the **Original Mailbox Address** box, enter the email address for the shared mailbox. There is not a dynamic content object for it, but it should resolve from the Global Address List as it did when you selected the mailbox in the first step.
17. In the **Mark as** box, select **Yes**. We want to mark this so that in the future if someone reviews the shared mailbox, they'll know this item has been read. However, this may not be enough, given that multiple people may access the mailbox and someone may accidentally change the status by just clicking on the message.

### Updated text values



In the preview version of this action, the dropdown featured the values *Read* and *Unread*. However, after the update, it is now a *Yes/No* choice. The documentation ([https://docs.microsoft.com/en-us/connectors/office365/#mark-as-read-or-unread-\(v3\)](https://docs.microsoft.com/en-us/connectors/office365/#mark-as-read-or-unread-(v3)))) lists it as the Boolean value *IsRead*. Choosing *Yes* can be understood to mean *Mark as IsRead? Yes*.

18. Add a **New step**. Search for the **Flag email (V2)** action and select it.
19. In the **Message Id** box, select the **Message Id** dynamic content object.
20. In the **Original Mailbox Address** box, add the shared mailbox email address again.
21. Under **Flag Status**, select **Complete**. This will help shared mailbox users visually verify that the message has indeed been processed:

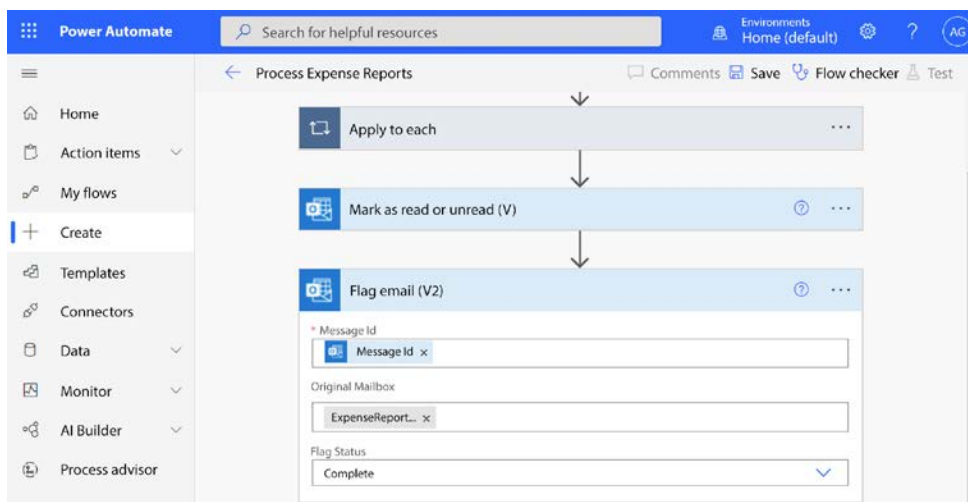


Figure 3.11: Configuring the email flag status



22. Click **Save**.

Time to test it out! Send a message to the shared mailbox meeting the requirements (an attachment and a subject including the text of *expense report*). After you've submitted an email message, verify that the flow has completed successfully.

## Verifying completion

To ensure that things have completed successfully, you'll want to review the output and verify it. You can use these steps to verify this particular flow:

1. From the Power Automate web portal (<https://flow.microsoft.com>), select **My flows**, select the ellipsis next to the flow, and click **Run history**:

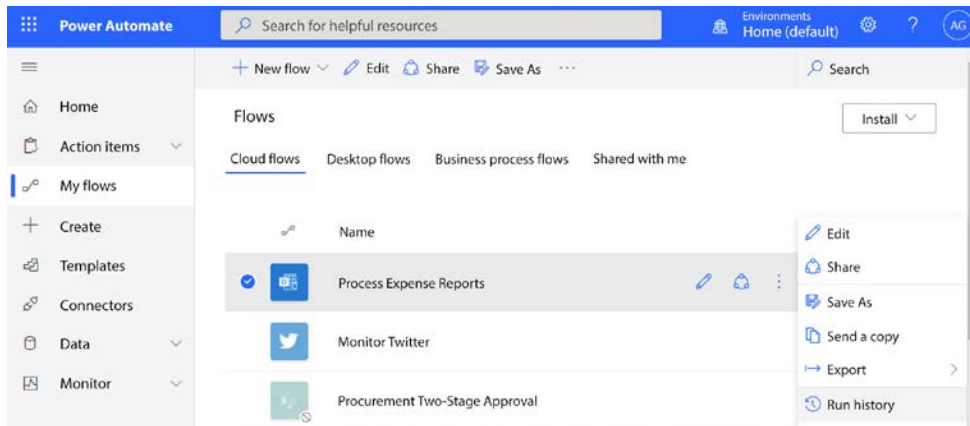


Figure 3.12: Preparing to view the run history

2. Select the individual run by clicking on the date.

3. Once the run data loads, you can click on the title bar of each step to expand it. If you run into errors or unexpected output, expand each step to verify the input values that Power Automate used during processing:

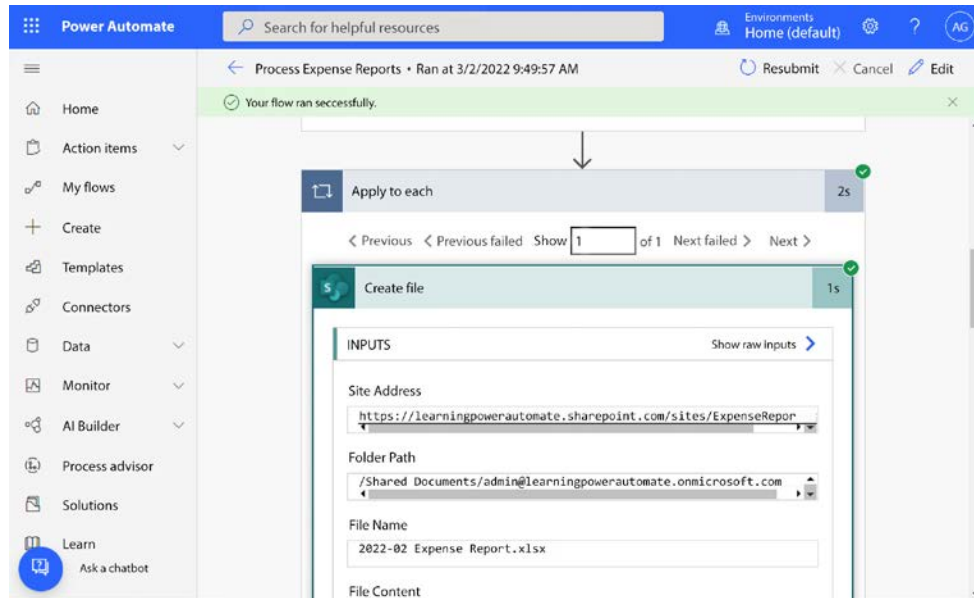


Figure 3.13: Expanding a run step

4. To check the connected SharePoint site, navigate to the SharePoint site containing the document library that you specified when creating the flow. Verify that the folder structure and file exist:

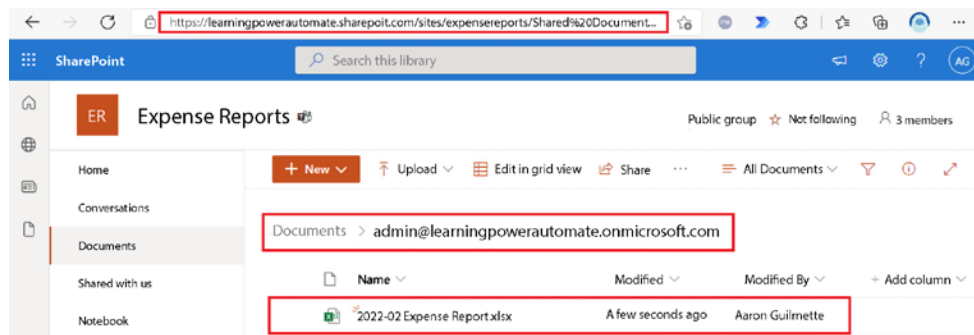


Figure 3.14: Verifying that the file has been placed

- Next, you can open the shared mailbox that the flow was configured to use. Look for the message and verify that it has been marked as read and the flag column shows a checkmark:

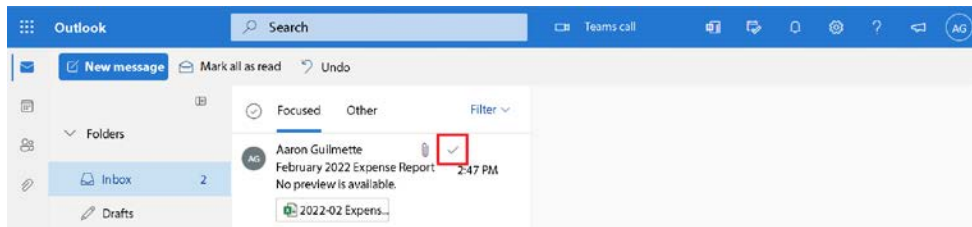


Figure 3.15: Reviewing email to ensure it was processed

At this point, you can be confident that your attachment handling flow is operating correctly.

Next, we'll finish up the configuration by modifying the flow to send a confirmation message.

## Sending email

Since sending things to a shared mailbox may seem like a bit of a black box, many organizations choose to send some sort of a confirmation email to let the sender know that the message has been received and processed. Sending an email is also a common task for Power Automate, and can be used in a number of circumstances, such as generating an email-based alert based on certain conditions or responding to a sender, as we're going to do in this example. The process is the same in either situation.

For this task, we're going to update the Expense Report attachment flow and reply with a message that the sender's file was received:

- Navigate to the Power Automate web interface (<https://flow.microsoft.com>) and select **My flows**.
- Select the previously created **Expense Report** flow, and then select the pencil icon to edit it:

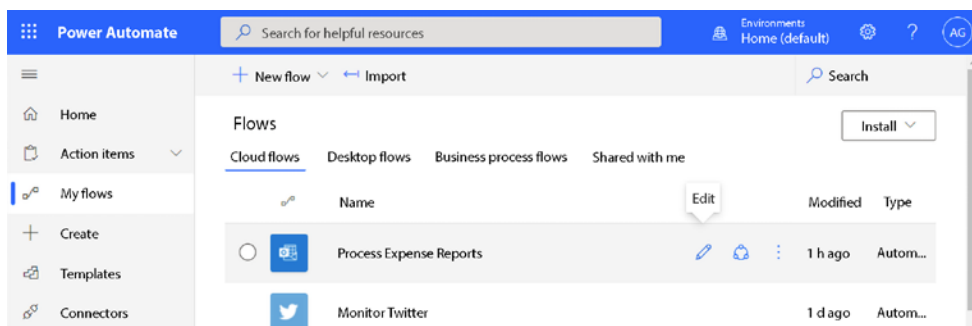


Figure 3.16: Editing an existing flow

3. At the end of the flow, select **New step** to add a new step.
4. Since the Expense Reports flow is using a shared mailbox, search for and add the **Send an email from a shared mailbox (V2)** action:

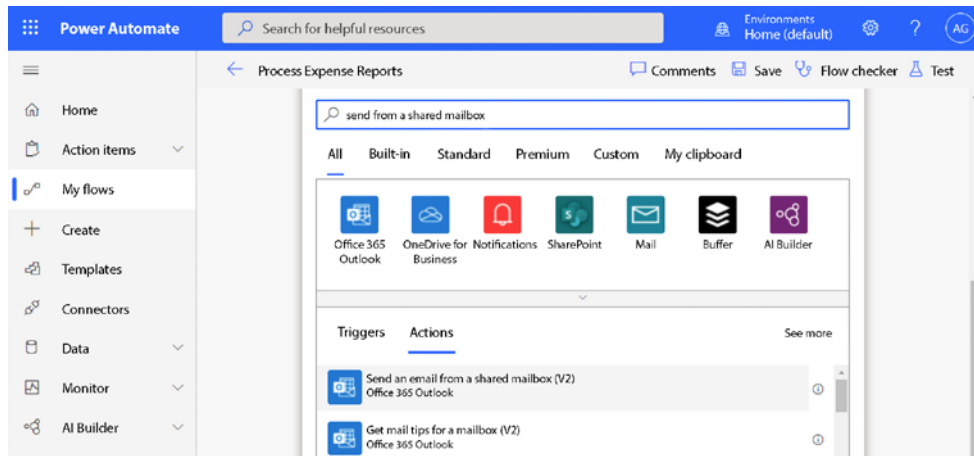


Figure 3.17: Adding a new send email action

5. In the **Original Mailbox Address** box, enter the value for the shared mailbox that will be generating the outbound message.
6. Under the **To** box, click the **Add dynamic content** button:

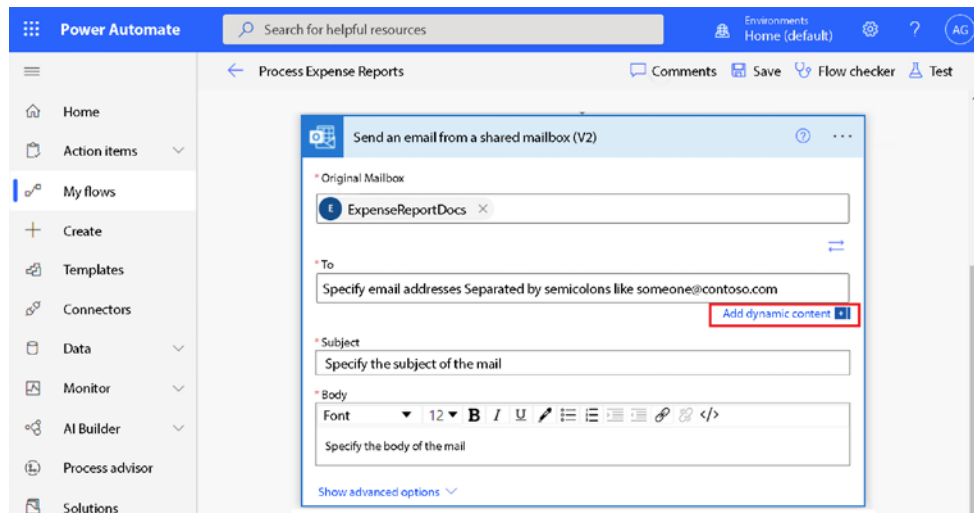


Figure 3.18: Expanding the dynamic content options

7. Under the **When a new email arrives in the shared mailbox (V2)** section, select **From** to use the address of the original sender:

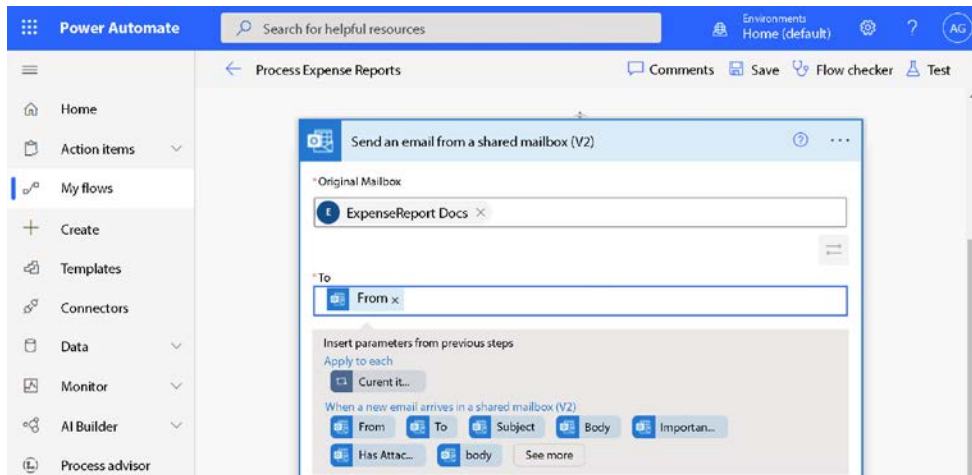


Figure 3.19: Adding the From field dynamic content token

8. In the **Subject** field, add a value that will be used as the subject of the outgoing message.
9. In the **Body** field, add any applicable text. You can use dynamic content objects here, if desired, as well as the rich text editor to change fonts or text properties.
10. If you want to set additional options such as fixed **Cc** or **Bcc** members, additional attachments, or message importance, select **Show advanced options**.
11. Scroll to the bottom of the message and click **Save**.

At this point, the flow has been modified to send back a response message upon successful receipt and processing. These changes will take effect the next time the flow is run.

When modifying or performing iterative development on a flow, you may want to use the option to resubmit or retry the last run with its parameters. You can use the test icon to specify options (such as resubmitting the last job), as shown in *Figure 3.20*:

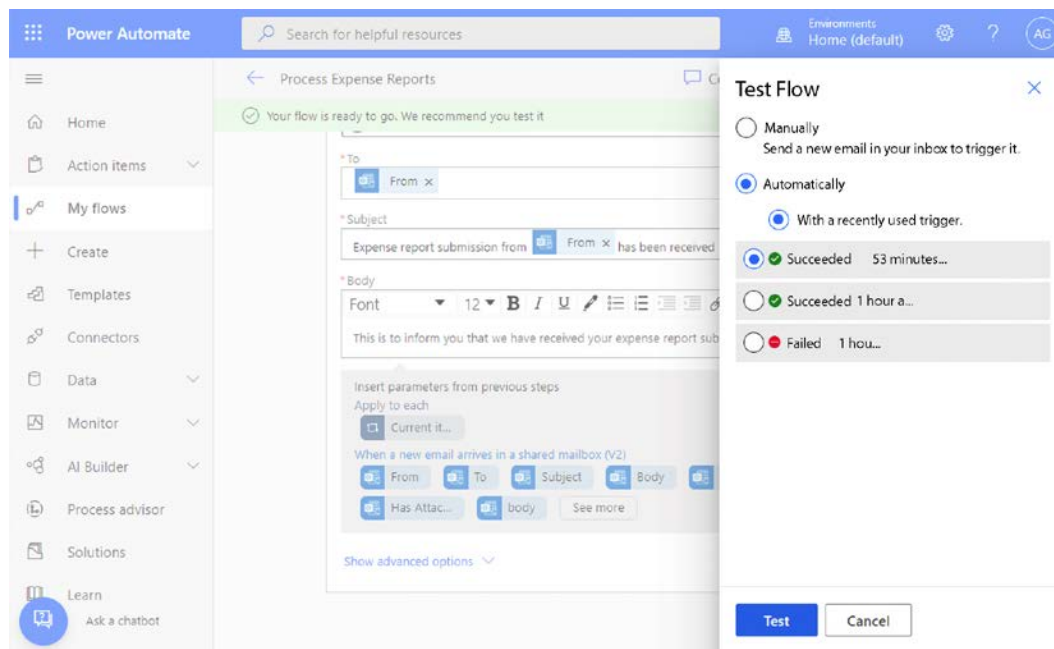


Figure 3.20: Testing the updated flow

It's important to note that the run history shows the steps for the last completed run. When you re-submit, it will run the flow in its current state, so any new steps added will be included.

You can check the sent items of the shared mailbox or the inbox of the original sender's mailbox to see the email confirmation.

## Summary

This chapter built on some of the foundational concepts in previous chapters and introduced several new concepts, such as connecting to mailboxes, saving attachments, and how the **Apply to each** function is automatically inserted to help iterate through multiple items in a step.

Using all of these tools together allowed you to build a simple flow that can process inbound messages and their attachments, and then respond to the sender that their message was received.

In the next chapter, we'll build on the knowledge you gained about SharePoint actions and use Power Automate to copy files between storage locations.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 4

## Copying Files

In the previous chapter, you learned a few basic concepts for automating email tasks, such as receiving, sending, and handling attachments. Frequently, a work process requires that you copy or move files between locations in order to make data available to new groups of users or to integrate the files into another business process. For example, another flow may need to interact with a saved email attachment or a notification may need to be sent, stating that a file has been received.

Processing files as part of a flow can mean interacting with the file itself, as well as its properties. Power Automate provides us with the functionality to manipulate a file's properties, its contents, and the object as a whole.

In this chapter, we're going to focus on some basic file-related concepts:

- Learning about file connectors and actions
- Working with files

By the end of this chapter, you should be familiar with common file connectors and actions and be able to use them to copy data, both inside the Microsoft 365 environment and to external storage locations.

### Learning about file connectors and actions

In *Chapter 3, Working with Email*, you saw that Power Automate can work with files as attachments to emails. Work processes that require us to automate file attachments may also benefit from automating further processing with those files. Files may need to be copied or moved, uploaded to a new service, ingested into another platform, archived, parsed, analyzed, or require approvals. Learning how to reference files and their properties is an important building block to building more complex flows.



Here's a quick rundown of some of the file storage mechanisms that are available to use:

- SharePoint Online
- OneDrive for Business
- OneDrive
- Azure Files
- Google Drive
- Box
- Dropbox
- FTP/SFTP

Most organizations will use Power Automate primarily within the context of SharePoint Online and OneDrive for Business. However, most organizations also have access to or use other file-hosting and sharing mechanisms as part of their manual business processes when working with partners, customers, or vendors, so it's important to be able to interact with as many platforms as possible. Many organizations, especially those focused in engineering, manufacturing, **Electronic Data Interchange (EDI)**, or logistics fields, may depend on legacy systems that require the FTP or SFTP protocol as well. Power Automate's file-handling capabilities can be used in many of those situations.

File connectors typically have the following types of trigger actions available, allowing you to create automated flows:

- When a file is created
- When a file is modified
- When a file's properties are modified

Similar to the email connector actions, file connectors allow you to perform the following types of actions (depending on the connector and what APIs the connected service has available):

- Update a file
- Copy a file
- Get the metadata of a file
- Delete a file
- List files or folders
- Create or delete folders
- Extract an archive file

- Create file-sharing links
- Check in or check out a file

Finally, the connectors allow you to extract, manage, or manipulate the content and metadata properties of the files, such as the following:

- The name or `DisplayName` of a file or folder
- The unique ID of a file or folder
- The file path
- The `LastModified` date and time
- The `MediaType` of the file

In addition to being able to use the available metadata of the files themselves, you may be able to include dynamic properties from other parts of the flow, as well as including them as part of your file-handling procedure.

Building on the concepts of dynamic content that we saw in *Chapter 3, Working with Email*, we're going to introduce a new concept: **expressions**. While dynamic content is used in the context of Power Automate to describe content properties derived from object metadata, expressions can be used to perform further processing in programmatic methods. If you are familiar with using Microsoft Excel formulas, then you'll notice that the concepts and formatting are similar.

**Expressions** are frequently used to calculate and format values. We're going to use an expression (or formula) to create custom folder names as part of working with files in the next section.

## Working with files

Files, from the perspective of Microsoft Power Automate, are objects that can be both the product and result of a workflow (as we saw in *Chapter 3, Working with Email*, when we looked at an expense report flow) or moved around through one. In this section, we're going to build on the expense report flow from the last chapter and copy the file from the expense report document library to another SharePoint site for archival purposes.

We'll also use Power Automate with another business scenario: publishing content to an outside storage service.

## Copying files to SharePoint

Suppose your organization has a requirement to make an archival copy of all the content that passes through a particular work process. To demonstrate how to do this, we're going to build on the flow we created in the last chapter in order to save email attachments and copy those file attachments to a new subfolder. We're going to perform a series of steps to calculate a unique name for the folder:

1. Extract the email's received time and date value
2. Use an expression to format the date as *yyyy-MM-dd* (four-digit year, two-digit month, two-digit day)
3. Use the result of the expression as part of a folder name to indicate when the file was processed

To complete these activities, you'll need a SharePoint site with a document library, as well as a folder inside the document library. We'll use a new SharePoint site called *Docucenter* in our example, but you can also use the site where you are saving the expense report files if you don't have access to another.



Depending on what features are available in your tenant, you may need to enable or disable Power Automate's Experimental Features' order to be able to add or edit expressions. If you need to disable Experimental Features', click the **Settings** icon (gear) in the upper-right hand corner of the web portal, select **View all Power Automate settings**, and toggle the slider for **Experimental Features**. In our tenant, we had to turn Experimental Features *on* in order to expose the ability to edit expressions.

You may also experience differences in capabilities exposed in the user interface depending on the zoom level of your browser. If dynamic content and expressions options aren't appearing, you might want to try adjusting the browser's zoom setting.

Follow these steps to update the flow:

1. Launch the Power Automate web portal (<https://flow.microsoft.com>) and select **My flows**.
2. Select the ellipsis next to the **Expense Reports** flow you created in *Chapter 3, Working with Email*, and click **Edit**.

3. Click the title bar of the **Apply to each** step to expand it:

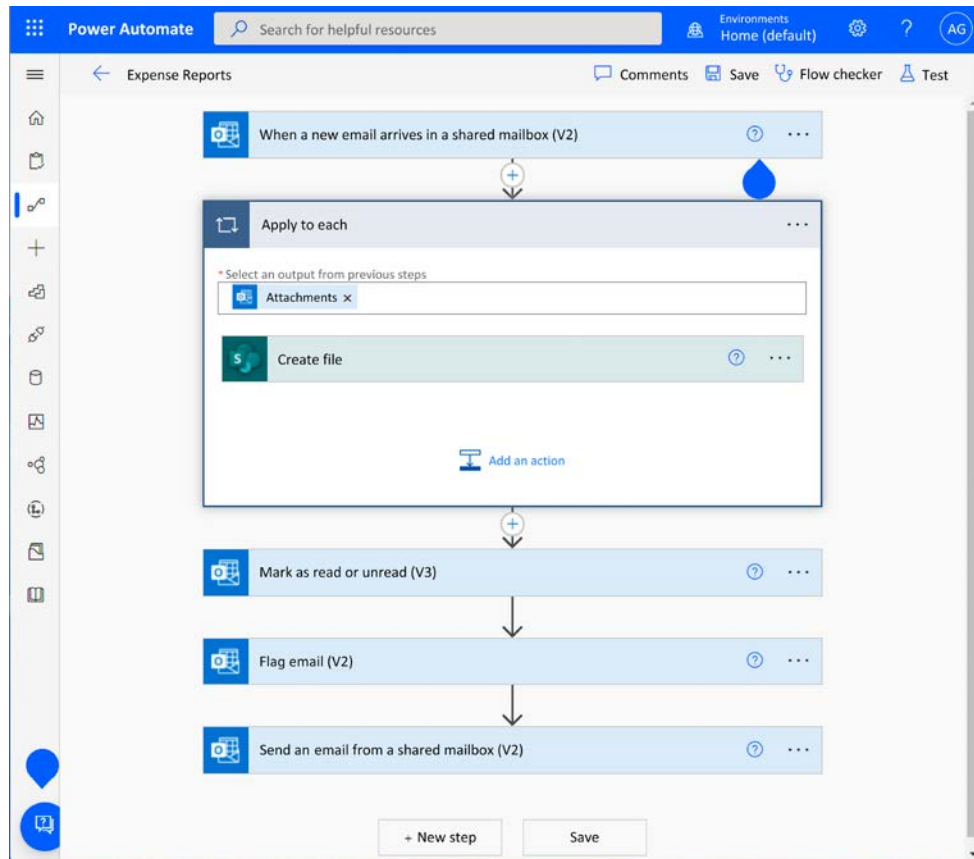


Figure 4.1: Expanding the step

4. Inside the **Apply to each** step, click **Add an action**.
5. In the **Search connectors and actions** box, type **SharePoint create new folder** and select the **Create new folder SharePoint** action.
6. In the **Site Address** box, enter the destination SharePoint site. If you're using an existing site, it may be visible in the dropdown. If you haven't used this site in a flow before, you may need to enter it manually.
7. In the **List or Library** box, select a document library that you wish to store the files in.

8. In the **Folder Path** box, enter a folder structure. You can browse for one, if you aren't going to customize it. In this case, we're going to use an existing folder called Archive, but then create a new folder path based on the date of the received email. Enter the top-level folder path with a trailing forward slash (/) character.



If you click the folder to browse the document library, you'll notice a folder named Shared Documents. However, when browsing the SharePoint site itself, the page will refer to the folder's display name, Documents. They are the same folder.

9. Select **Add an expression**:

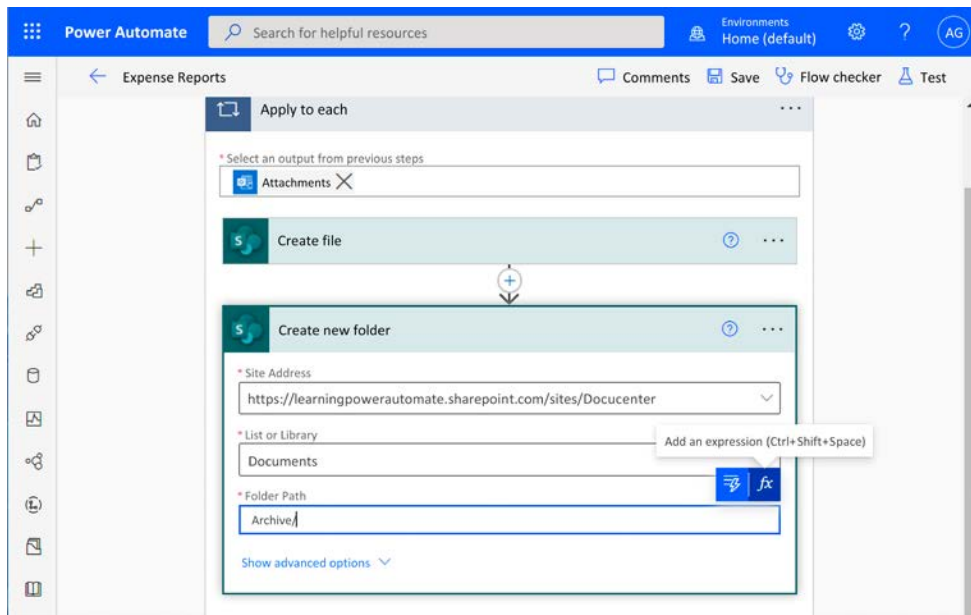


Figure 4.2: Creating a new folder in the step

10. In this example, we're going to create a folder based on the date/time the message was received and use that as part of the folder name. Begin typing date and select the **formatDateTime** function. Then, enter trailing parentheses (()), as shown in the following screenshot:

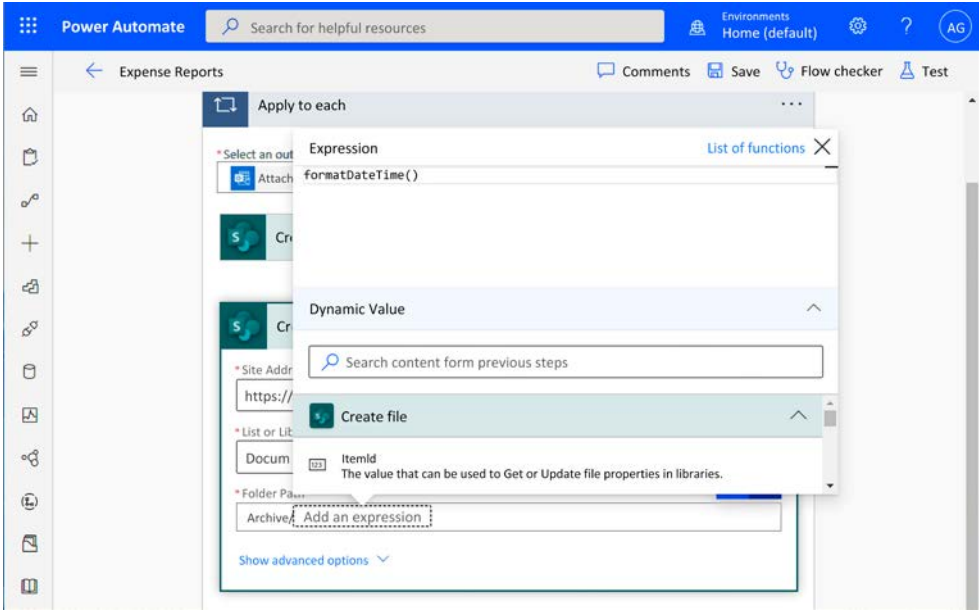


Figure 4.3: Adding the `formatDateTime()` expression

11. In the **Dynamic Value** search box, search for the **Received Time** dynamic value content object from the **When a new email arrives in a shared mailbox (V2)** action and select it:

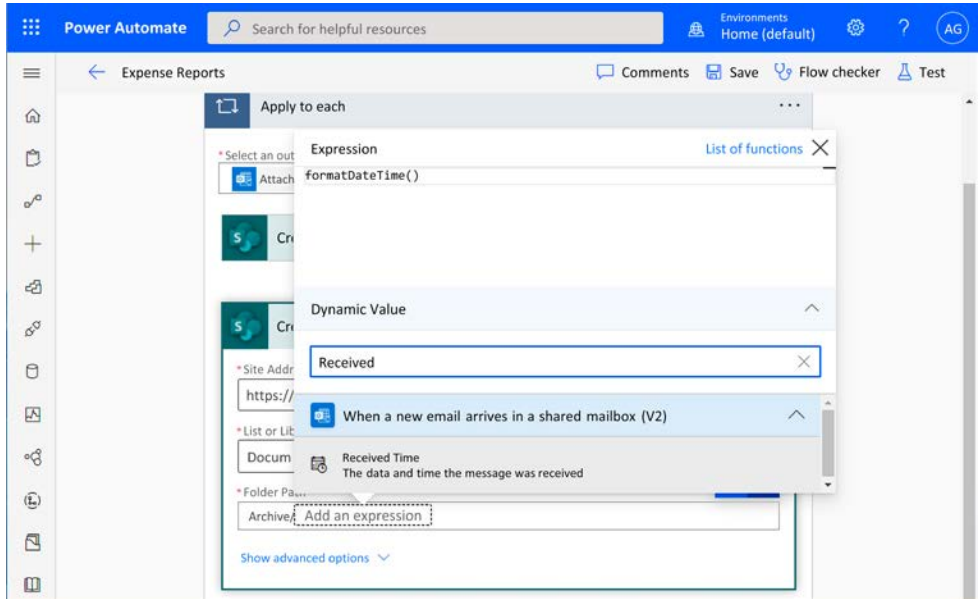


Figure 4.4: Selecting the **Received Time** dynamic value

12. The **Expression** box value will update. Move the cursor to the end of the line between the closing square bracket (]) and the closing parenthesis ()) and add , 'yyyy-MM-dd'.
13. The expression in the box should now look like:

```
formatDateTime(triggerOutputs()?[ 'body/receivedDateTime' ], 'yyyy-MM-dd')
```

Click outside the **Expression** dialog box when finished.

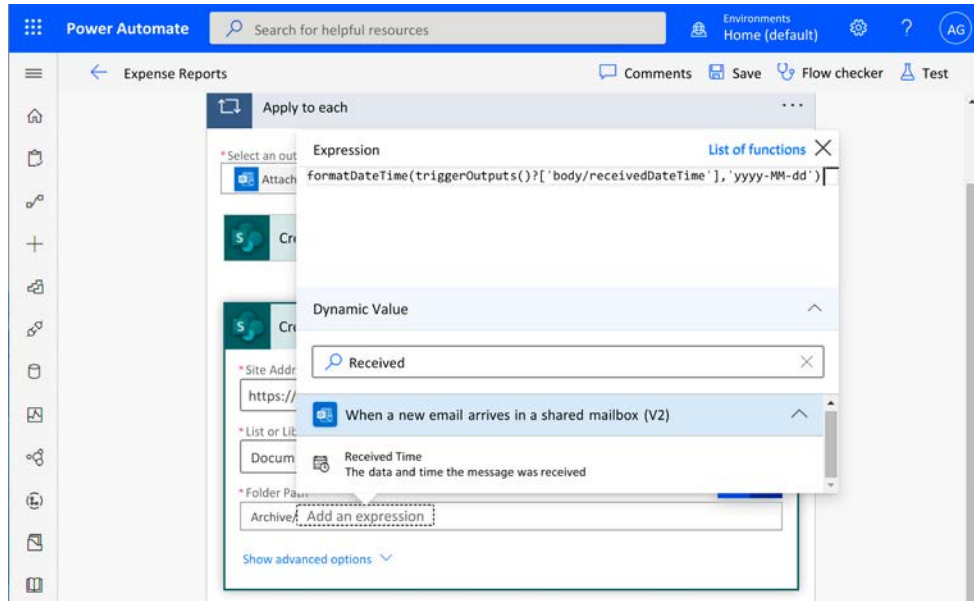



Figure 4.5: Editing the expression

14. Hover over the expression in the **Folder Path** box to ensure that it saved correctly.
15. Click the **Add an action** button after the **Create new folder** action.
16. In the **Search connectors and actions** box, type **SharePoint copy file** and select the **Copy file** action from the list.

In the **Current Site Address** box, select the existing SharePoint site where the expense reports flow is configured to save the file.

- 17. In the **File to Copy** box, select the dynamic value content object that represents the **Id** of the file saved in the **Create file** action.



There are two very similar dynamic value tokens: **ItemId** and **Id**. Choose the one that is labeled **Id**.

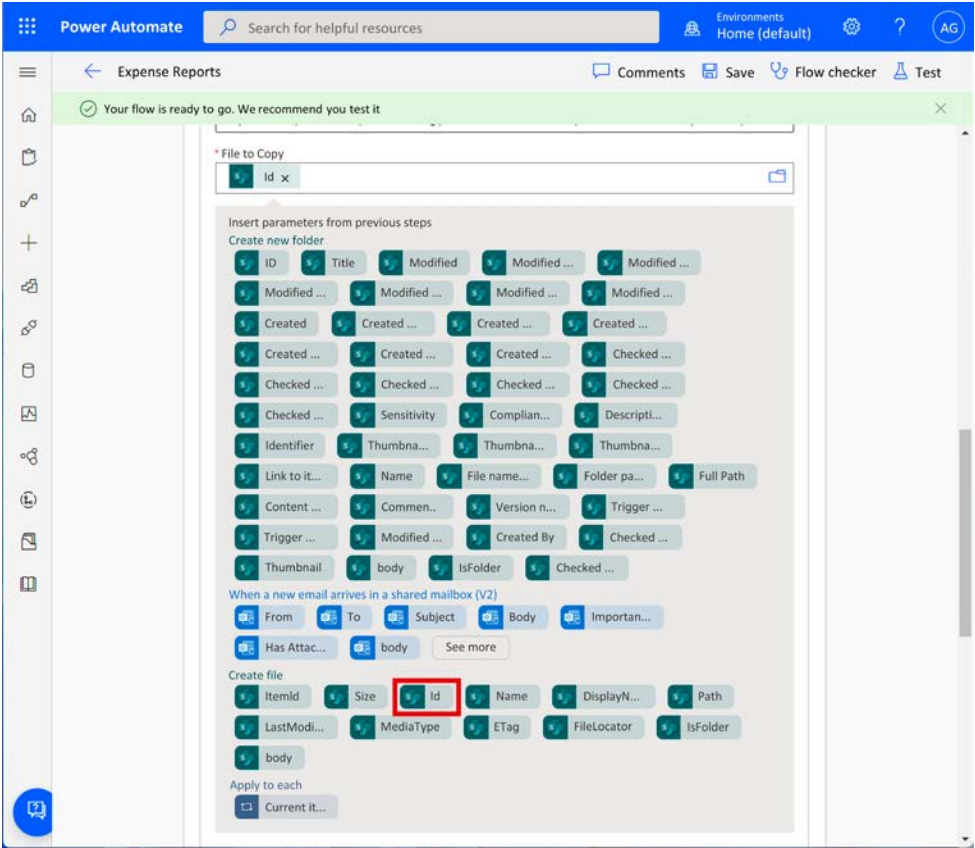


Figure 4.6: Selecting the identifier that represents the file to be copied

- 18. From the **Destination Site Address** dropdown, select a SharePoint URL where you wish to save this new archival copy of the file. If you haven't used this URL in a flow before, you may need to type it in manually.



19. From the **Destination Folder** dropdown, select the **Folder path** dynamic value token from the **Create new folder** action. Then, select the **Name** dynamic value token. Alternatively, you can recreate the expression that you used to create the folder earlier. Do *not* insert a ' / ' character between the values.

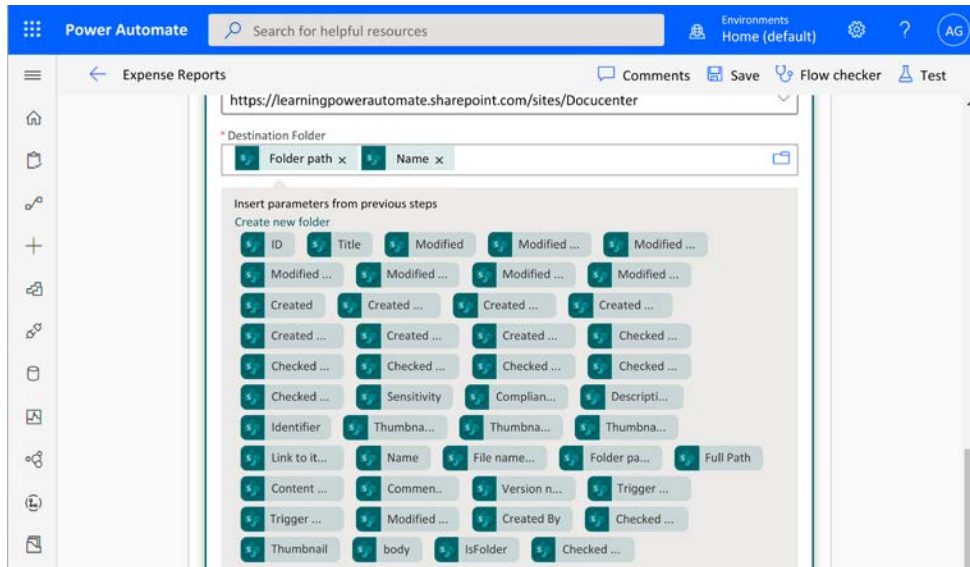


Figure 4.7: Configuring the destination folder path

20. In the **If another file is already there** box, you can choose from four options: **Copy with a new name**, **Fail this action**, **Replace**, and **Enter a custom value**. Which option you choose depends on your business requirements. In this case, we're going to select the **Copy with a new name** option. You can use the **Enter a custom value** option to build your own naming convention using expressions and dynamic content, as we did in the previous step, as well.
  21. Click **Save** to save the updated flow.
- Next, we'll verify the results of the changes.

## Verifying results with Test Flow

It's important to make sure that the syntax is correct for when we make any updates to configurations. To do so, you can use the **Test Flow** function at the top of the page. Follow these steps to do so:

1. Click **Test** to test it and ensure that you get the expected result. You can choose to trigger this manually or reuse data from a previous run. In this case, we're going to use the results from a previously successful run. Select the run and click **Save and Test**:

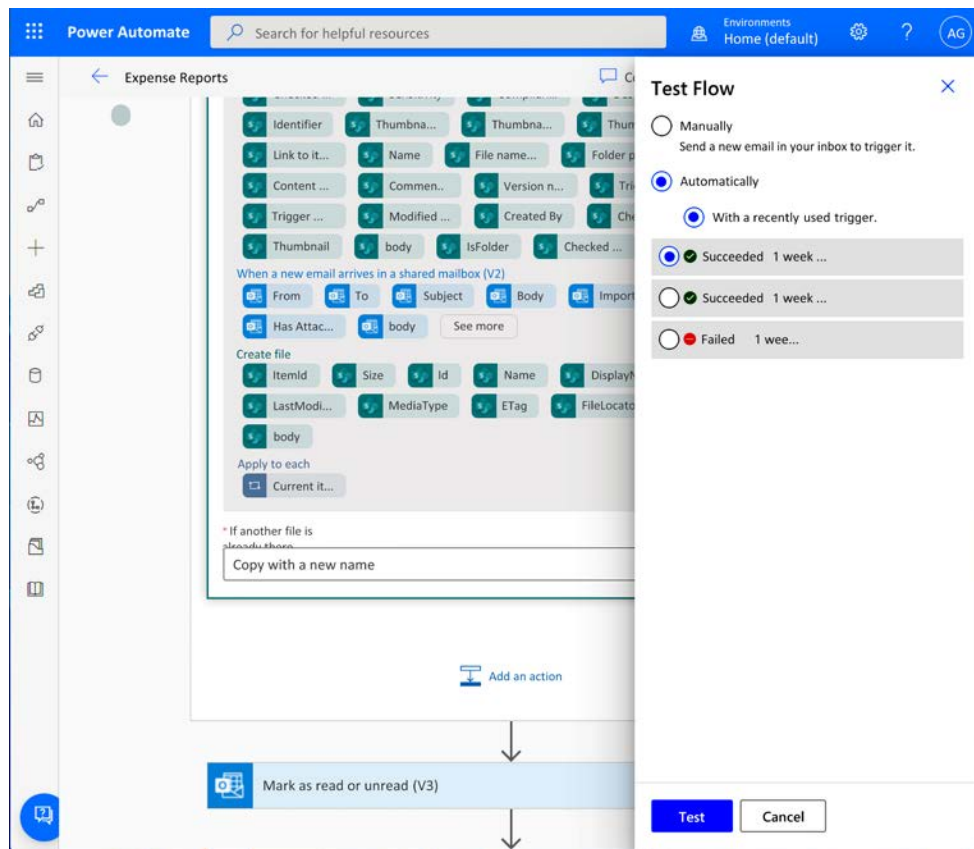


Figure 4.8: Testing the updated flow

2. The results should come back successful. If not, review the flow configuration and make any necessary updates:

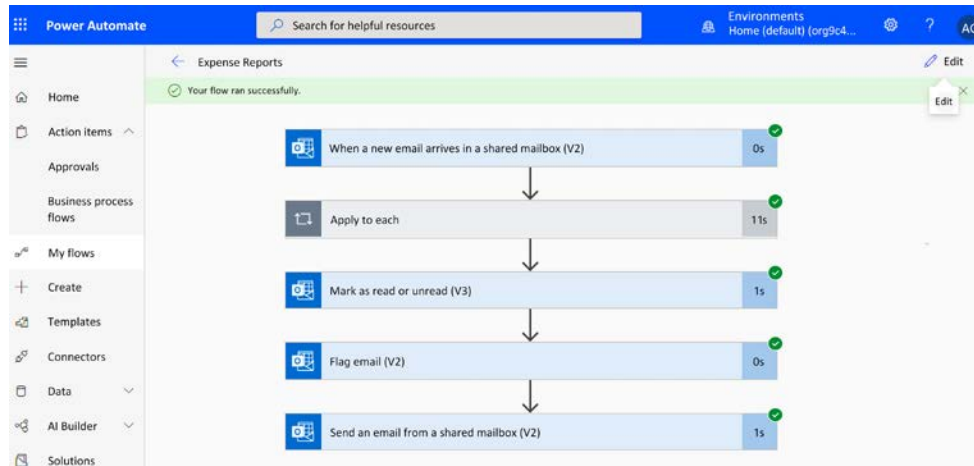


Figure 4.9: Verifying the results from a test run

While seeing a success banner is a good sign, it's also important to verify that the output is as expected. We'll look at this next.

## Verifying the results manually

While the Test Flow function may state that it was successful, that doesn't necessarily mean it's *correct*. It just means that it completed without *errors*. When it comes to formatting expressions or functions, many are case-sensitive. If, for example, you are processing a date such as *March 2, 2022* and you use `YYYY-MM-DD` instead of `yyyy-MM-dd` in the `formatDateTime` expression, you'll get a folder named `YYYY-03-DD` instead of `2022-03-02`.

To verify that you achieved the expected result, you can expand and examine the resulting JSON values at each of the steps. Alternatively, you can simply navigate the target library, as shown in the following screenshot:

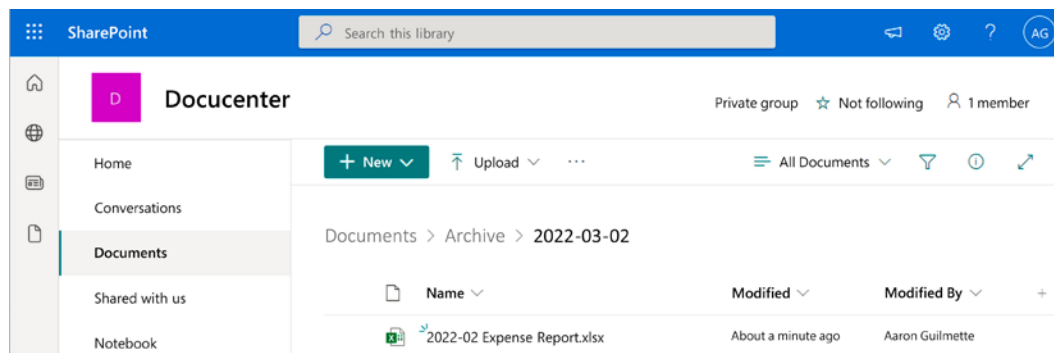


Figure 4.10: Verifying the results manually

With that, the file that was attached to the recently processed expense report will be deposited into the correctly named folder.

Next, we'll look at using the Dropbox connector to work with files.

## Publishing files to Dropbox

Another popular file copy scenario is moving files to repositories *outside* of SharePoint Online. The public site option for SharePoint Online was deprecated several years ago, which makes it more difficult, in some instances, to make information broadly available.

For example, suppose your organization hosts a SharePoint Online site where individuals save **Requests for Proposals (RFPs)** that have been prepared to go out to bid, and you need to make those files automatically available on a public download site for potential vendors. You can create a flow to monitor that SharePoint Online site and copy contents to the Dropbox site to automate this task.

If you want to follow along with the steps for this flow, you'll need a SharePoint site to monitor as well as a Dropbox account. You can sign up for a free Dropbox account at <https://www.dropbox.com>.



You may want to disable the Experimental Features view in Power Automate for this series of steps.

To get started, follow these steps:

1. Launch the Power Automate web portal (<https://flow.microsoft.com>) and select **Create flow**.
2. Select **Automated cloud flow**.
3. Enter a name for the flow (in this case, we'll use Publish RFP) and search for the SharePoint trigger called **When a file is created or modified in a folder**. Select it and click **Create**:

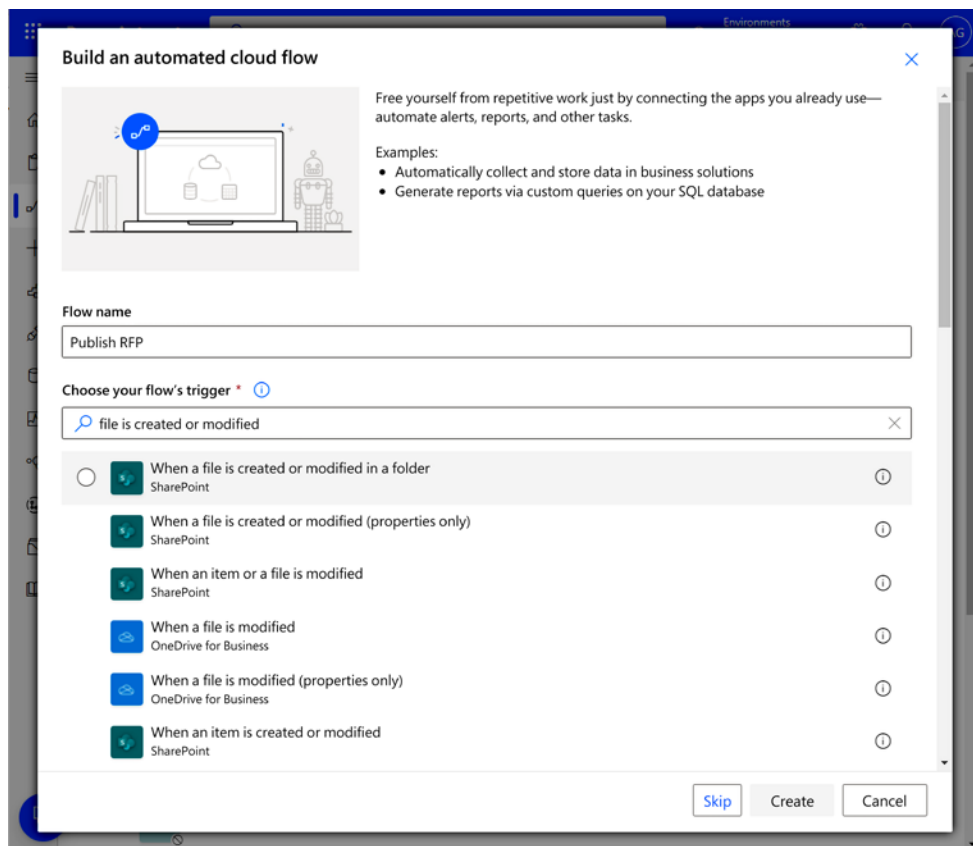


Figure 4.11: Creating a new cloud flow

4. In the **Site Address** box, enter the URL for the SharePoint site that will be monitored for new files. In this case, we're going to be monitoring a site called *PublishedRFPs*.
5. In the **Folder Id** box, select the folder that will be monitored. Select the folder icon to begin browsing, and then select either the folder name to choose the current folder or the arrow to choose a subfolder. See *Figure 4.12*:

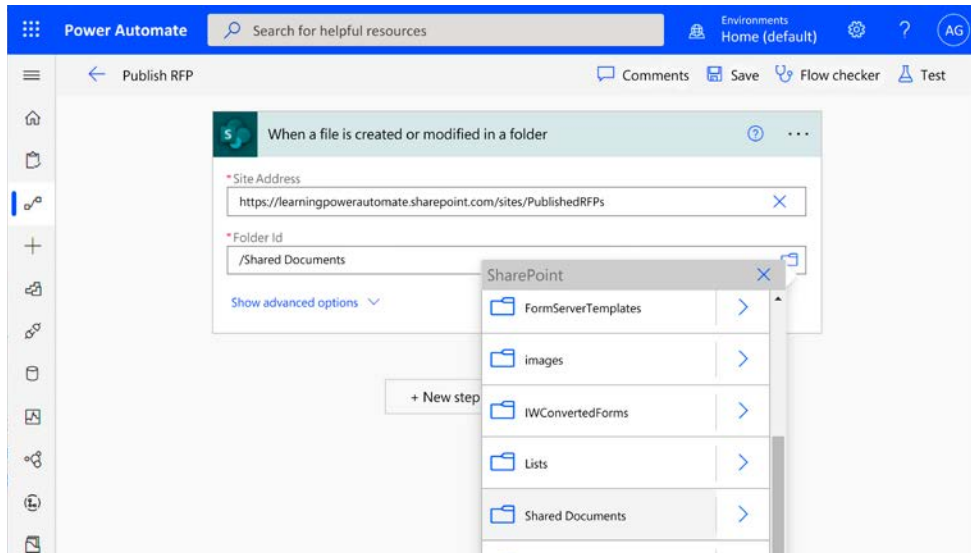


Figure 4.12: Selecting a folder to monitor



Remember, the folder name **Shared Documents** refers to the default document library on the SharePoint site.

6. Click **New step**.
7. Then, select the **Choose an action** box, search for Dropbox, and select the **Create file** action.
8. Click **Sign in** and provide the necessary credentials for the service. If a CAPTCHA or other identity challenge is presented, provide it.

9. Once authentication has been verified, in the **Folder Path** box, click the folder icon and browse to the target folder where the file will be saved:

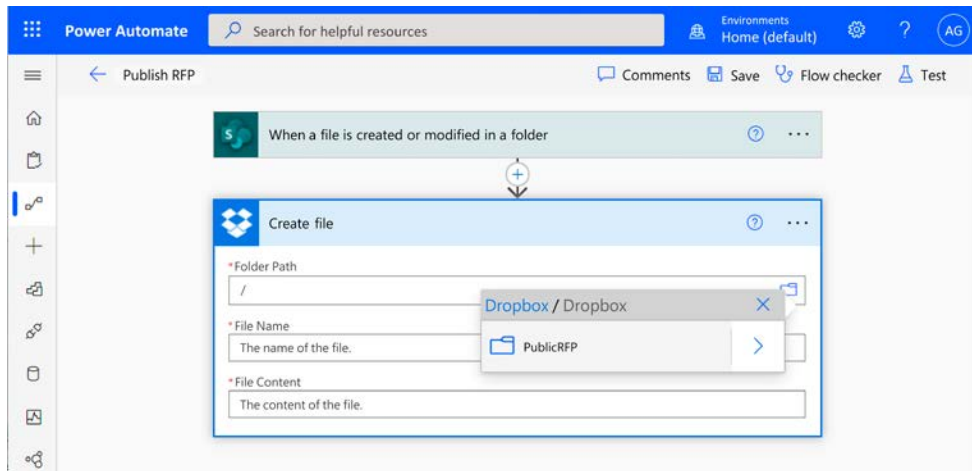


Figure 4.13: Selecting the target folder path

10. In the **File Name** box, select the **x-ms-file-name-encoded** dynamic value.
11. In the **File Content** box, select the **File Content** dynamic content item.
12. Click **Save**.

At this point, the flow is ready to be tested.

## Verifying the results

To verify the functionality of this flow, follow these steps:

1. Upload a document to the monitored SharePoint document library:

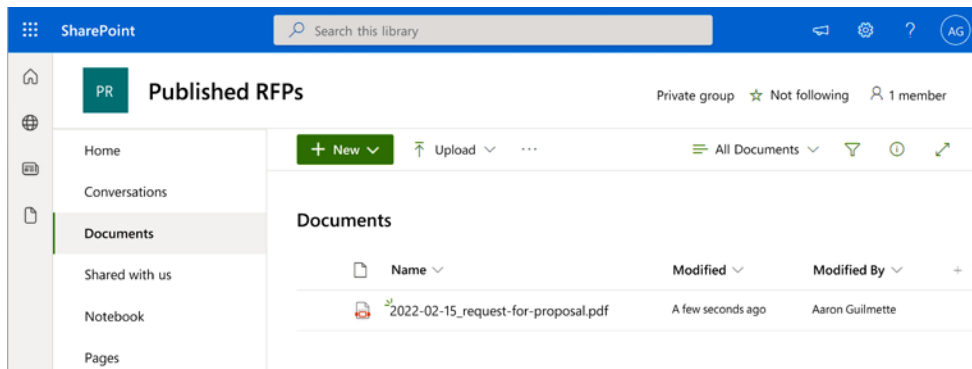


Figure 4.14: Uploading a test file to the monitored SharePoint folder

2. Next, from the Power Automate web portal, navigate to **My flows**.
3. Select the ellipsis next to the newly created flow and select **Run history**.
4. Click the date under the **Start time** column for the most recent run:

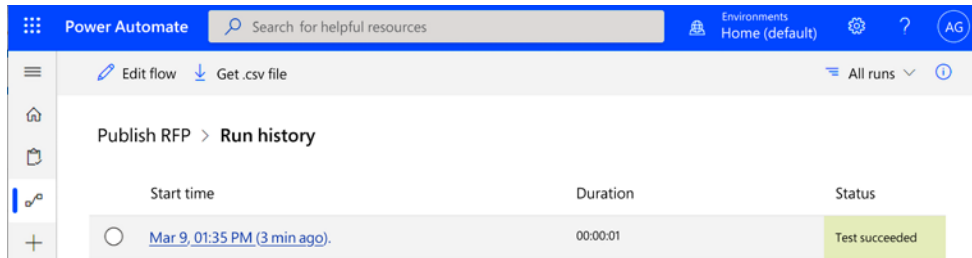


Figure 4.15: Selecting the run to view

5. Click the title bar of a step to expand its details:

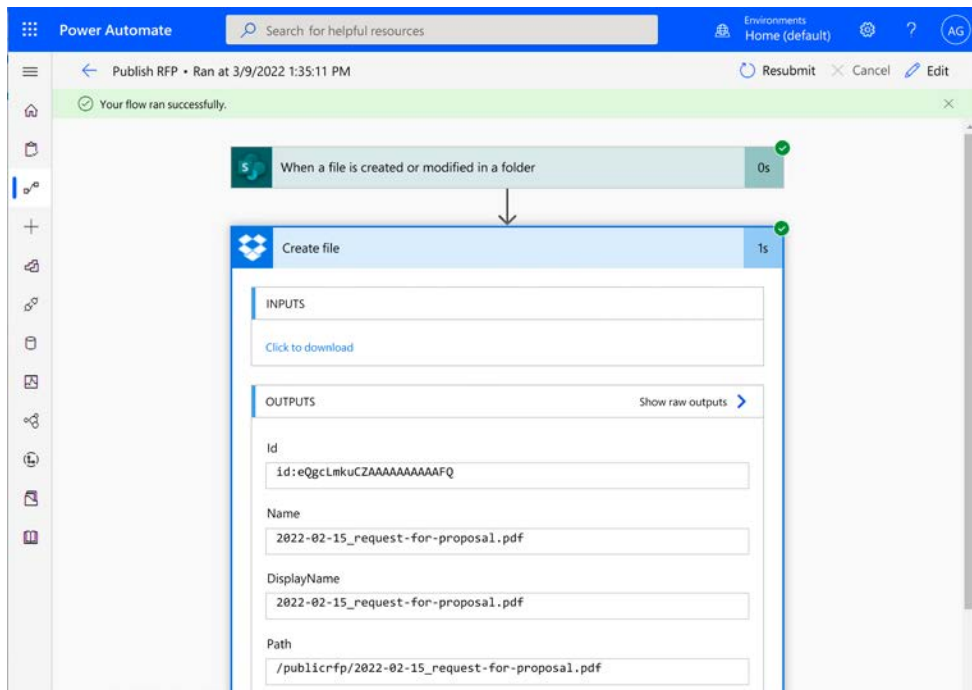


Figure 4.16: Viewing the run details



- Next, log into the Dropbox account and select **Recents** to review the recent files:

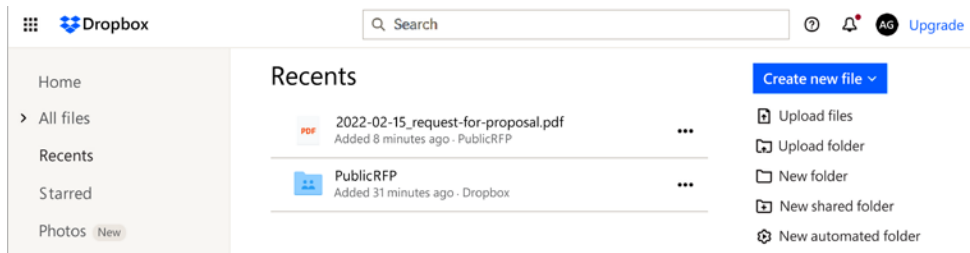


Figure 4.17: Reviewing recent files in Dropbox

- Close the browser window.

At this point, you can be confident that your publishing flow is working correctly.

## Summary

This chapter introduced a few new actions (*Create folder* and *Copy file*) for the SharePoint connector, a new connector to an external service (Dropbox), as well as the concepts of functions and **expressions**. Expressions can be used to further create custom content, calculate and format values, or extract data from entities.

Using these new components, you were able to copy files internally between SharePoint sites, as well as copy them to an external service. Both types of flows (copying files internally and externally) can be used as part of solving very common business automation problems. Increasing your familiarity with expressions will help you make detailed flows with customizable data output.

In the next chapter, we'll shift the focus to creating instant or button flows.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 5

## Creating Button Flows

Up to this point, we've worked primarily with *automated flows* – that is, flows that happen based on Power Automate detecting, or being notified of, a change and then acting upon it. In this chapter, we're going to shift gears to create a different type of flow – one that happens when you manually start it.

Since the original launch of Power Automate as Microsoft Flow, *button flow* has been known by a few other names, including *manual flow*, *instant flow*, and now *instant cloud flow*. While this terminology is largely interchangeable, the term *button flow* is typically used to indicate a flow that has been published to a mobile device that can be initiated by tapping a button in the Power Automate mobile app experience, while the terms *instant flow* or *instant cloud flow* are typically seen in the Power Automate web portal.

In this chapter, we're going to cover the following topics:

- Learning about button flows
- Creating a button flow to email a manager
- Executing a button flow from the Power Automate mobile app

By the end of this chapter, you'll be familiar with creating and using button flows. Let's get started!

### Learning about button flows

Button flows (or instant cloud flows), as previously mentioned, are manually triggered flows. They don't monitor anything, nor do they have any sort of REST-based trigger that instantiates them. Button flows can be used for a variety of low-impact tasks, such as executing reminders or notifying individuals.

A button or instant flow can be created in three different ways:

- From a button flow template
- From a blank template
- Imported from a package

One of the best ways to get started, especially when working with common tasks, is to look at the examples of others and templates. If you think you might be doing something that someone else has done before, a template is a great place to start. You can also import previously saved and exported flows (you'll learn more about this in *Chapter 18, Exporting, Importing, and Distributing Flows*).

Button flows can be created from either the Power Automate mobile app or the Power Automate web portal. While you can create complex flows on both the mobile device and web platform, it's generally easier (and quicker) to create them on a desktop computer. You can see examples of button flow templates in the following screenshot:

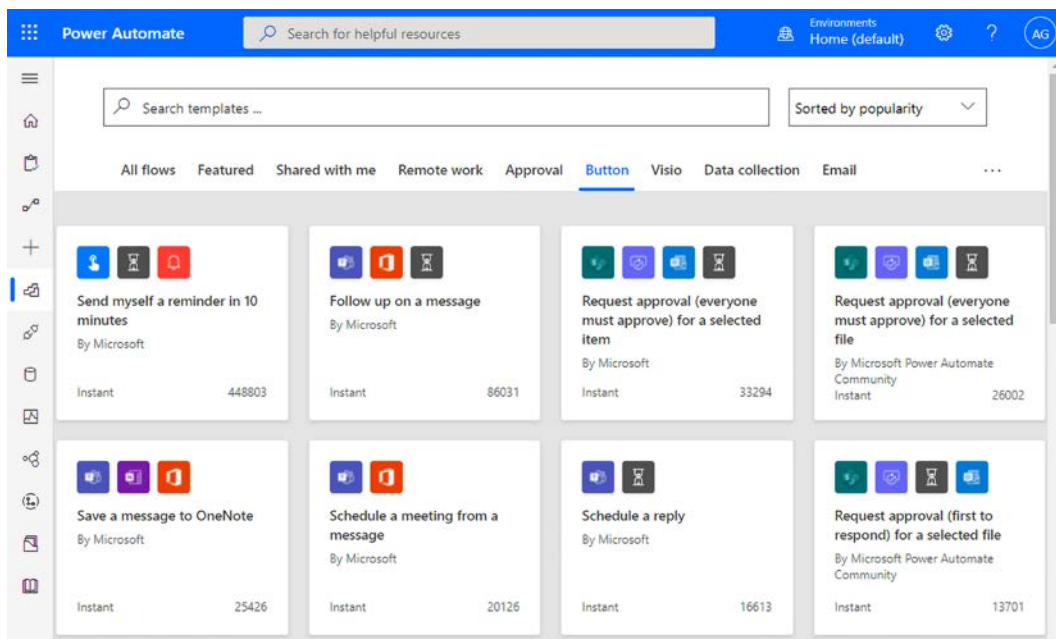


Figure 5.1: Button flow templates

While most button flows are designed to perform a one-time task with simplicity, more complex instant or button flows can be used to interact with data from SharePoint lists, documents, or data stored in Dataverse (formerly known as Common Data Service).

In the next section, we'll create a button flow from the web portal.

## Creating a button flow to email a manager

There are times when you just need a quick form or letter to respond or send someone a notification, such as when you're out of the office due to a holiday or illness. In this section, we're going to configure this simple button flow to email your manager that you're going to be out of the office. Like the flows you've created in the last few chapters, we're going to take advantage of dynamic content to make sure we retrieve the value stored in the manager property for an account in Office 365.

In order to do this, we're going to introduce a new connector and some interesting actions:

- The **Office 365 Users** connector: This connector allows you to connect to Azure Active Directory to retrieve information pertaining to the user object data. You can learn more about the **Office 365 Users** connector here: <https://docs.microsoft.com/en-us/connectors/office365users/>
- The **Get my profile (V2)** action: This action will retrieve your specific details from Azure Active Directory. You can learn more about what properties this action can access here: <https://docs.microsoft.com/en-us/graph/api/resources/user?view=graph-rest-1.0#properties>
- The **Get manager (V2)** action: This action retrieves the manager property of the selected user. In order to use this, the **Manager** Exchange property must be configured. You can learn more about the **Get manager (V2)** action here: [https://docs.microsoft.com/en-us/connectors/office365users/#get-manager-\(v2\)](https://docs.microsoft.com/en-us/connectors/office365users/#get-manager-(v2))

With these new components, we'll be able to retrieve your manager metadata value and send off a pre-populated email. As you work through the example, take careful note of the order of the steps when working with Office 365 data – in this case, retrieving your profile data and then retrieving the manager value.

We'll create the flow in the Power Automate portal and execute it on a mobile device – that way, you can interact with the flow on multiple platforms.

To perform these steps, you'll need to have an Exchange Online mailbox that has been configured with a manager property. Most corporate environments have already set this through synchronization with Azure AD.

Follow these steps to complete the example:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **Create** and select the **Instant cloud flow**.
2. Enter a name for the flow and select **Manually trigger a flow**. Then, click **Create**:

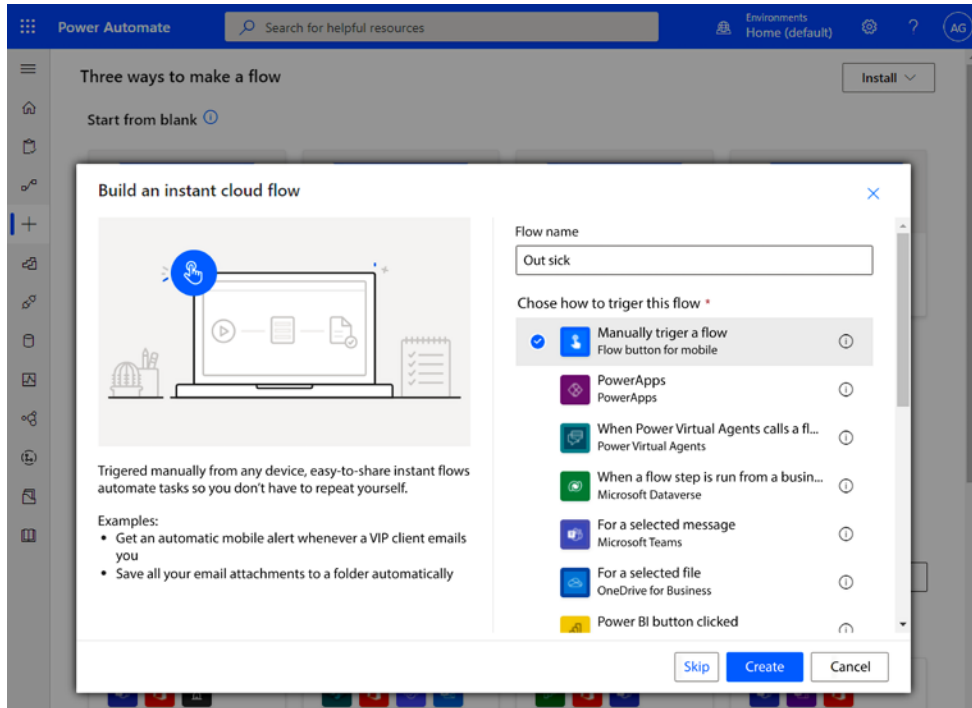


Figure 5.2: Creating an instant cloud flow

3. Click **+ New step** to add a new step:

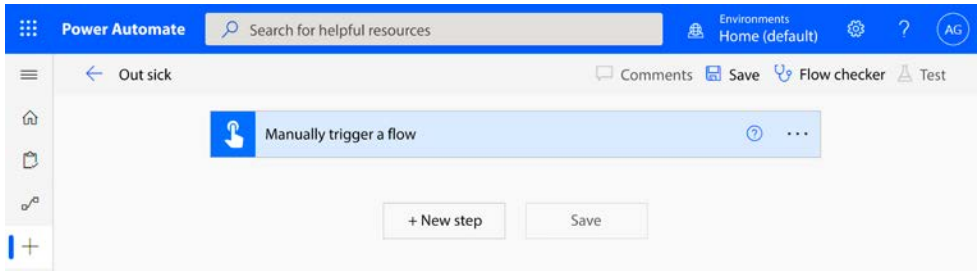


Figure 5.3: Adding a new step to the flow

4. In the **Choose an operation** search box, enter office 365 get my profile and select the **Get my profile (V2)** action:

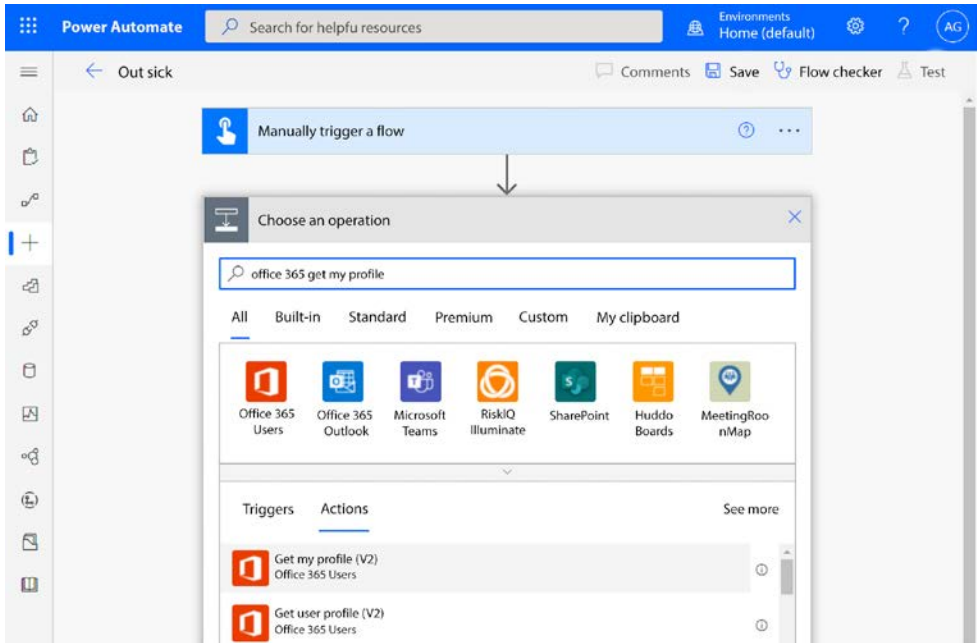


Figure 5.4: Adding the Get my profile (V2) action

5. If you haven't used or fully configured any of the Office 365 connector objects before, you'll need to provide authentication to use them. To do this, select the ellipsis for the **Get my profile (V2)** action, and then, under **My connections**, either select your existing connection (if it was detected) or click **+Add new connection**. Enter any credentials:

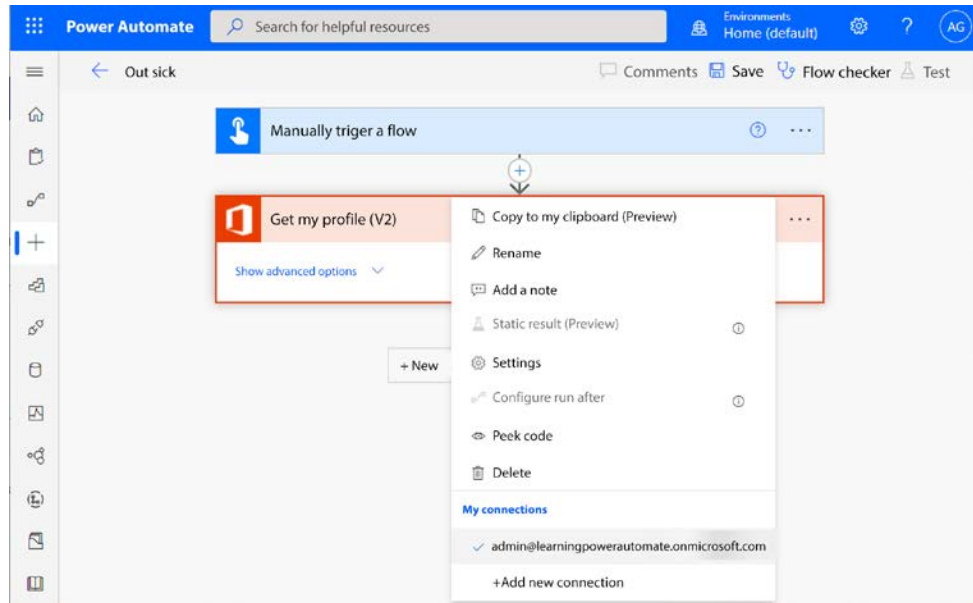


Figure 5.5: Configuring a connection to Office 365

6. Once your authentication is successful, click **+ New step** to add a new step.

- 7. In the **Choose an operation** search box, enter office 365 get manager and select the **Get manager (V2)** action from the list:

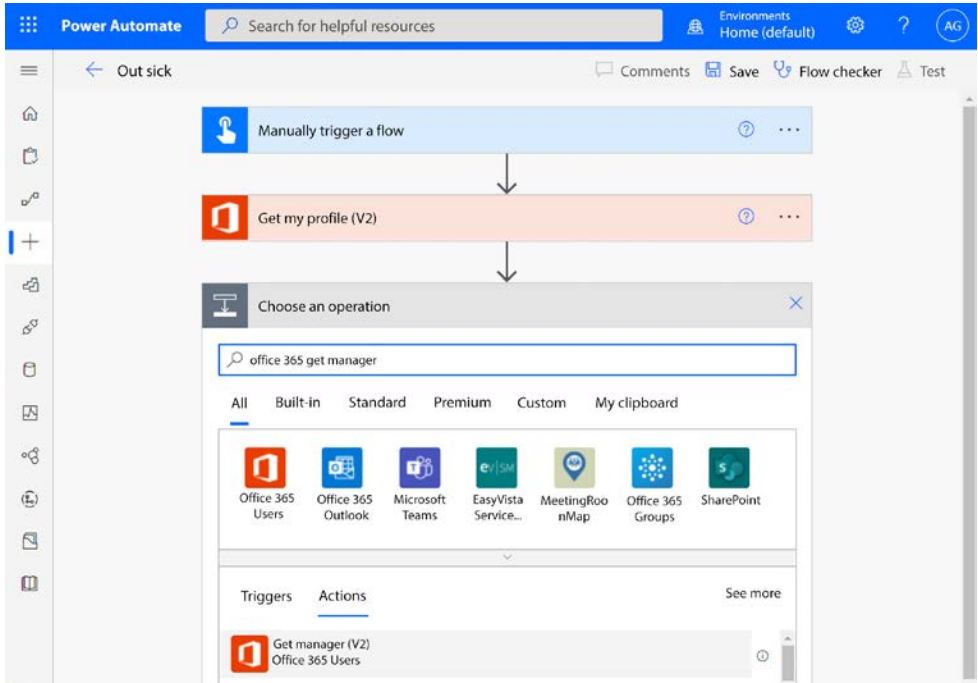


Figure 5.6: Adding the Get manager (V2) action



8. Click on the **User (UPN)** value box, and then select the **User Principal Name** value under the **Get my profile (V2)** section. This option will retrieve the `UserPrincipalName` property that was discovered as a result of the **Get my profile (V2)** action from earlier:

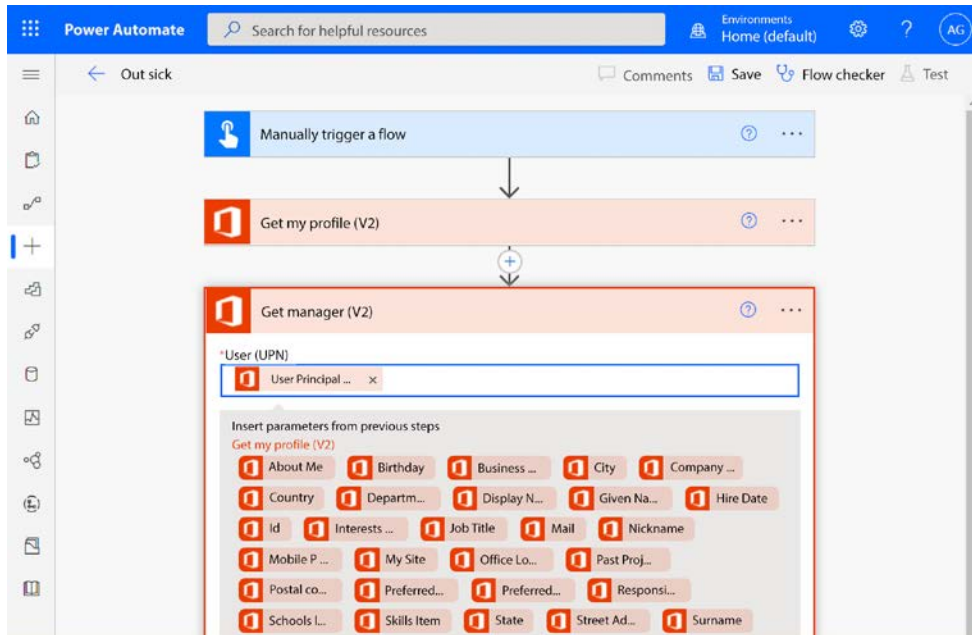


Figure 5.7: Selecting User Principal Name from the profile properties



Once you have added the dynamic value, **User Principal Name**, you can hover over it to see its configuration. It should say `outputs('Get_my_profile_(V2)')[ 'body/userPrincipalName' ]`, indicating that it is using the `userPrincipalName` value from the **Get my profile (V2)** action.

9. Click **New step**.
10. In the **Choose an operation** search box, enter **Office 365 Outlook Send Email** and select the **Send an email (V2)** action:

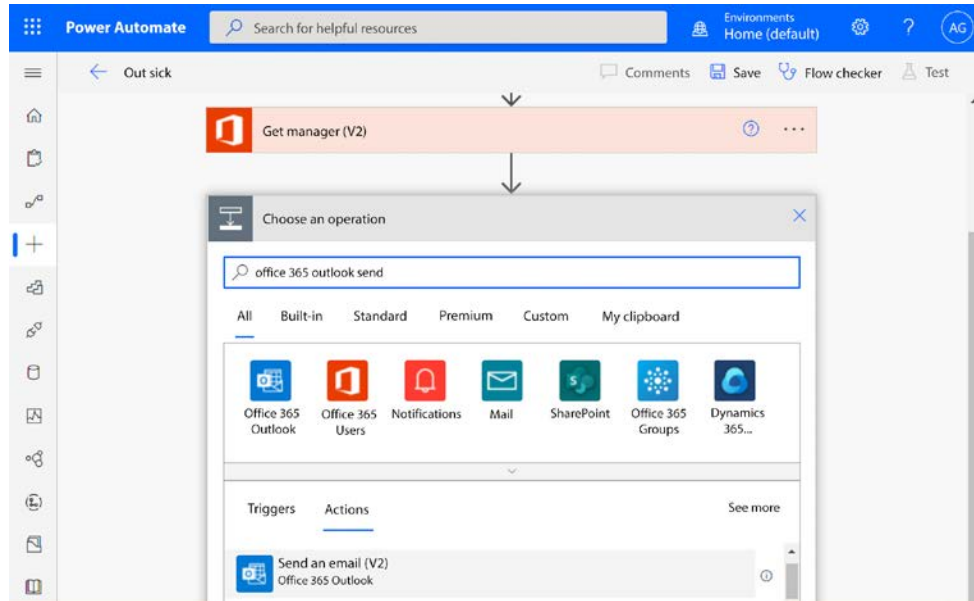


Figure 5.8: Selecting the Send an email (V2) action

11. At this point, you've retrieved the value stored in the profile for your manager. You can then use the dynamic value content object to populate the **To** address field. Power Automate is aware of the property type restrictions of actions and, by default, automatically selects the most appropriate or likely options. In this case, select the **Mail** property under the **Get manager (V2)** section (you may need to select the **Add dynamic content** button if the fields are not automatically displayed):

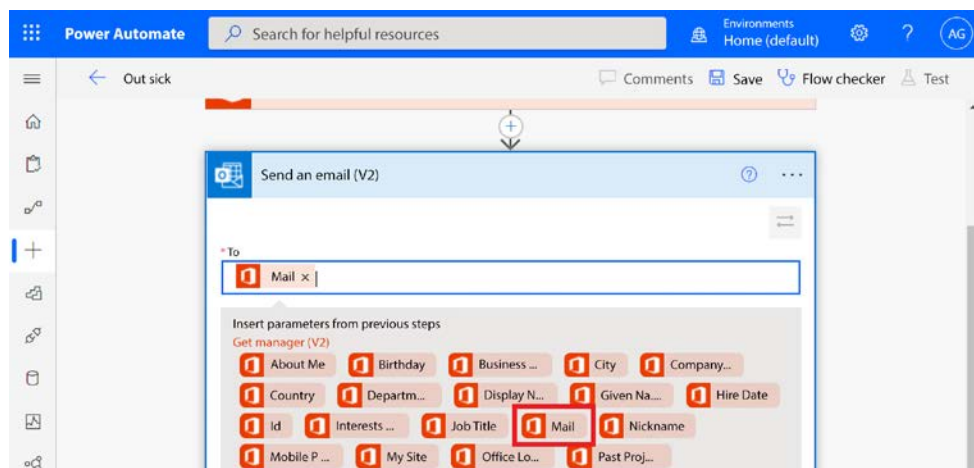


Figure 5.9: Selecting the Mail dynamic content value

12. Complete the flow by filling out the **Subject** and **Body** fields. This can be a good opportunity to experiment with new expressions. For example, one way you can retrieve today's date and format it in **month/day/year** format is by combining the `formatDateTime()` expression and the `utcNow()` expression. Select the `formatDateTime()` expression, and then supply the expression `utcNow()` as the first argument and specify the date format as the second argument: `formatDateTime(utcNow(), 'MM/dd/yyyy')`.



Remember, if you don't seem to have the ability to add expressions, you may need to adjust the UI by going back and toggling the **Experimental Features** switch in the Power Automate settings or by changing the zoom level of the browser.

To learn more about the `formatDateTime()` expression, see <https://aka.ms/formatdatetime>.

13. Verify that all of the fields of the action have been completed:

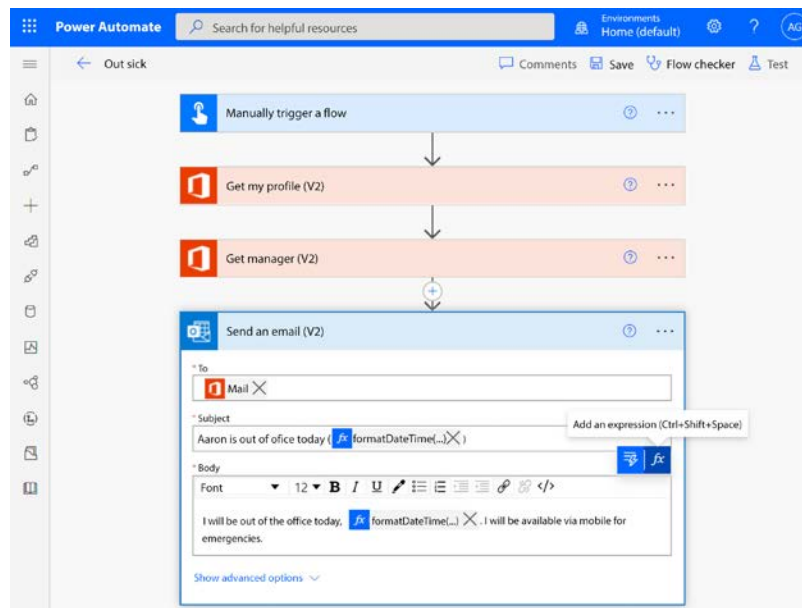


Figure 5.10: Composing an email with expressions

14. Click **Save**.

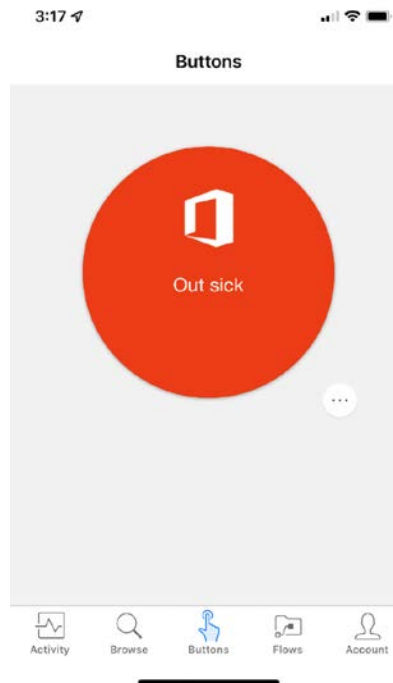
At this point, you should have a properly configured flow that will email your manager. You can view the flow in **My flows** and select the **Play** icon to launch it if desired.

However, since this is designed to be a button that you can execute from anywhere, we'll test it from the Power Automate mobile app.

## Executing a button flow

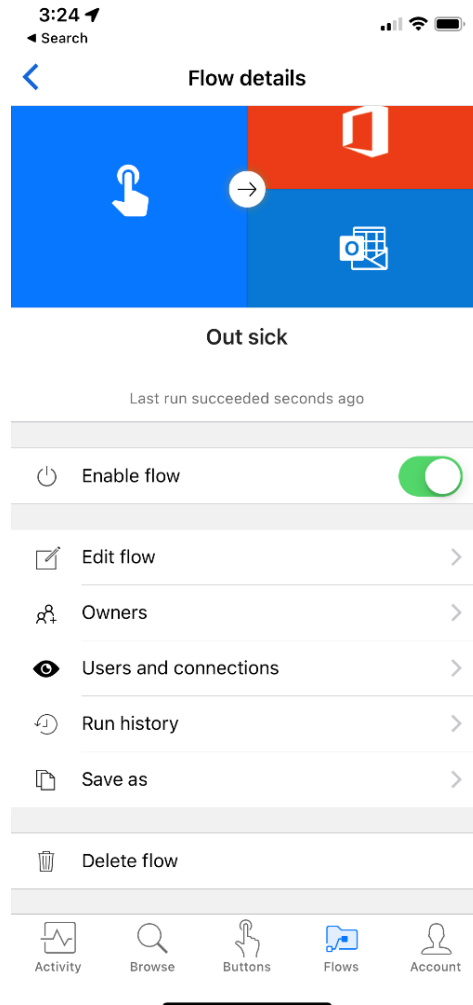
Once you have created a button flow, the best way to see its usefulness is by launching the Power Automate mobile app. Once you have signed in, follow these steps to execute the newly created flow:

1. Verify that the new button flow appears. After launching the Power Automate app, select **Buttons**. You should see your flow listed, as shown in *Figure 5.11*:



*Figure 5.11: Button flows*

2. Tap the button to execute the flow.
3. After it has completed, tap **Flows** in the tray and then tap the button flow to open the **Flow details** page, as shown in *Figure 5.12*:



*Figure 5.12: Examining the flow details*

4. Tap **Run history** to display the recent runs.

5. Select the run, and then examine the details to ensure that it was completed successfully. You can expand any of the steps to view the results of any step, including the expansion of dynamic content or the evaluation of expressions:

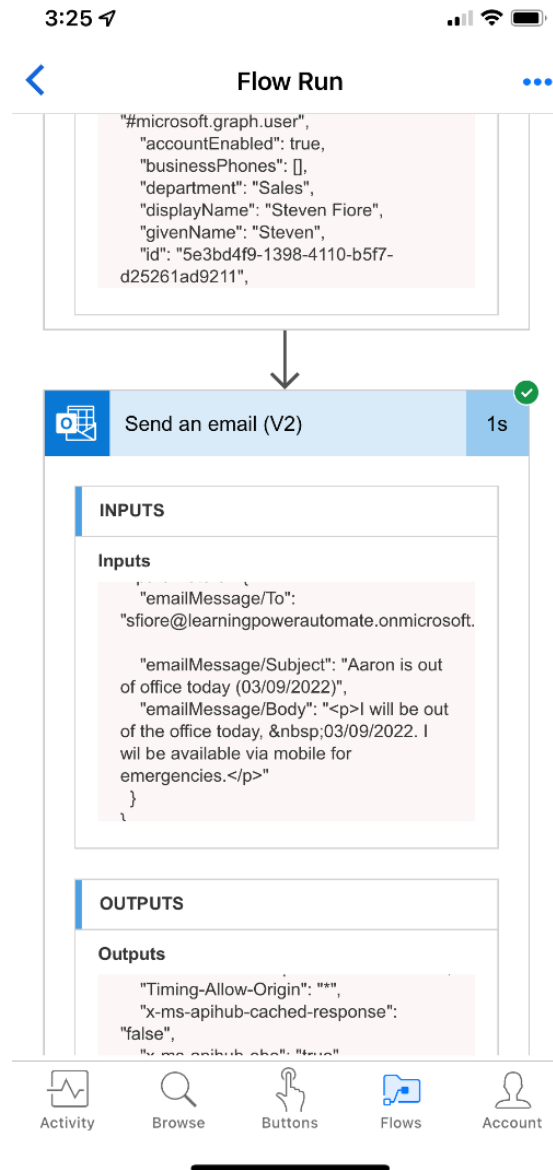


Figure 5.13: Viewing the individual steps in a run

The flow is complete. You can perform further verification, if desired, by checking the **Sent Items** folder of the sender's mailbox. Since the flow is authenticated, outgoing emails will be sent from, and saved in, the sender's mailbox.

## Summary

In this chapter, we covered new connectors and actions, as well as an example of how to nest an expression inside another expression by formatting the value from `utcNow()` with the `formatDateTime()` expression. By combining multiple expressions, you can display variable data in a format that makes sense for the flow.

As you've seen, button flows can be useful for quick tasks that can be executed from either the web portal or a mobile device. While button flows can be created from mobile devices, you may want to consider creating complex flows from the web interface so that you can see more components on the screen at the same time.

In the next chapter, we're going to look at using push notifications to increase visibility and draw attention to noteworthy items.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 6

## Generating Push Notifications

Push notifications are a type of communication mechanism used to deliver a message (typically a popup or banner) to an end user's device. Push notifications are a handy way to keep users connected to data and activities, drive engagement scenarios, or prompt users to perform either an action or a related app task.

Microsoft Power Automate enables users to configure their own push notifications as part of a flow. For example, in addition to processing an email attachment, you may wish for a flow to send a notification to your device when an email attachment has been processed.

In this chapter, we'll look at how push notifications work and how to integrate notifications into your flows. The topics we'll cover are as follows:

- Learning about push notifications
- Configuring a notification for emails from your manager
- Reviewing a notification

Using notifications effectively can help you stay on top of important tasks. Let's begin!

### Learning about push notifications

Push notifications are delivered using subscriber-based **Platform Notification Systems (PNSes)**. PNSes are unique to mobile platforms. Platforms include Firebase for Android devices, Apple Push Notification services for iOS platform devices, and Windows Notification services for Windows-based devices. Vendors that develop multi-platform applications must integrate with each PNS for which they wish to offer push notifications.



The following diagram depicts a basic overview of how a push notification works:

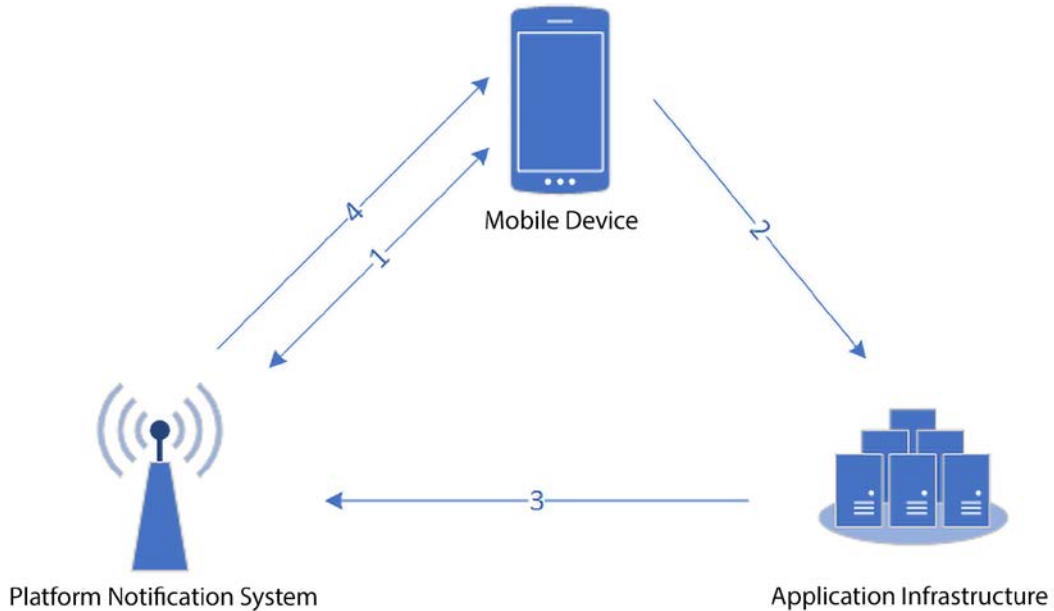


Figure 6.1: Overview of a working push notification

In order to implement push notifications, developers must write both the client and server applications in a way that can communicate with the PNS. When activated, the notification-enabled application contacts the PNS and requests a special token called a **handle**. This token identifies the application and individual device asking for a subscription to a particular service. The handle or token represents a unique pairing of a user's device and application. It cannot be used for any other application or device combination.

When a user configures an application to receive notifications, the application requests the handle from the PNS and then saves it to the application server's infrastructure. As events occur in the application infrastructure that trigger notifications, the application infrastructure contacts the PNS, supplying the handle to which the event or notification corresponds. In turn, the PNS uses the handle to contact the application on the target device.

In this context, we'll use the Power Automate mobile app to request a handle for a notification. When the conditions of the flow are met, the Power Automate service will communicate with the PNS to alert your mobile device with a notification.

Notifications are really just actions that can be configured whenever you need to raise an alert and can be delivered through several means, such as push notifications, email, a Teams activity feed, or a Power App. A notification action can be added anytime a set of conditions or circumstances require immediate attention by a user, whether internal or external to the Power Automate and Office 365 environments.

Next, you'll see how to configure a basic flow that integrates push notifications to a mobile device.

## Configuring a notification for emails from your manager

Configuring a flow that uses a push notification is very straightforward. You can add push notifications to existing flows or create a new flow that utilizes a push notification.

In this example, we're going to create a new flow that sends a push notification when the monitored mailbox receives an email from the user's manager. We're going to use the following components that we learned about in the previous chapter:

- **Connectors:** Office 365 Users and Outlook
- **Actions:** Get my profile (V2) and Get manager

We're also going to introduce a new connector and new actions:

- **Notifications connector:** This connector allows you to interact with the notifications service. You can learn more about the notifications connector here: <https://docs.microsoft.com/en-us/connectors/flowpush>.
- **Send me a mobile notification action:** This action sends a push notification to a mobile device. You can learn more about the Send me a mobile notification action here: <https://docs.microsoft.com/en-us/connectors/flowpush/#send-me-an-email-notification>.
- **Condition control:** This allows you to perform different actions based on a condition, such as the value of a property.

Depending on how your applications are configured, you may be able to configure notifications to take advantage of **deep linking**, in which the notification takes the user to the relevant item (as opposed to a dashboard of items). You can learn more about deep linking here: <https://powerapps.microsoft.com/en-us/blog/powerapps-deep-linking/>.

You'll also notice that we're introducing a new type of action – a **condition**. Rather than being tied to the capabilities of a particular connector or service, it can help your flow make decisions on what actions to perform.

## Introducing conditions

Conditions, as indicated, are a type of control action that allows you to build more complex flows based on the values of certain properties, expressions, calculations, or inputs. In traditional programming or scripting languages, you might see this referred to as an If statement. The following screenshot illustrates an empty condition action:

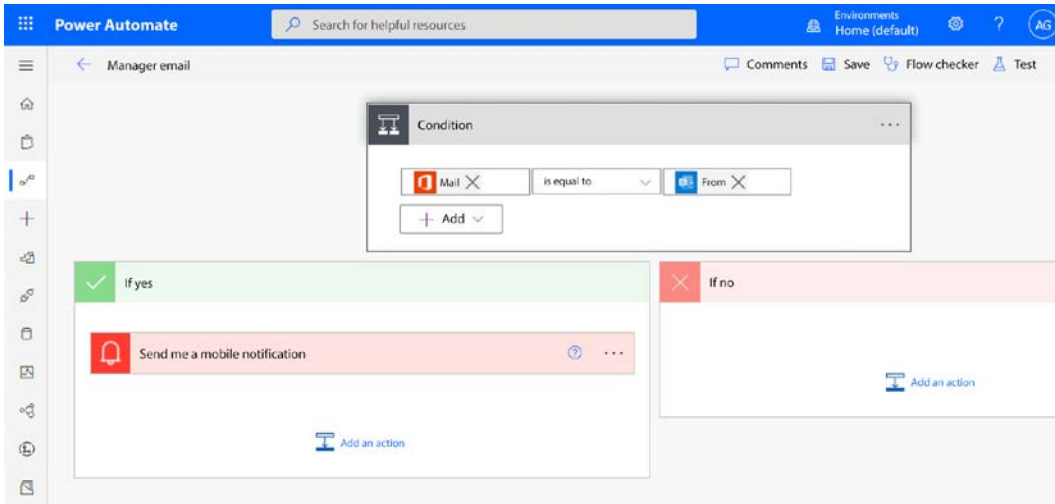


Figure 6.2: Condition example

In programming or scripting languages, the If construct allows the application to follow different paths (or branches) based on the result of a previous command or choice.

If statements typically follow one of these examples:

```
if [condition equals true], then [do action]
```

Or

```
if [condition equals true], then [do action];
else [do other action]
```

The condition control in Power Automate functions in much the same way. You can control the set of actions a flow will execute depending on the values present in the flow.

Power Automate conditions can perform several different types of evaluations, as shown in the following list of operators:

- **contains**
- **does not contain**

- is equal to
- is not equal to
- is greater than
- is greater than or equal to
- is less than
- is less than or equal to
- starts with
- does not start with
- ends with
- does not end with

When evaluating values against each other, you'll want to make sure you're using the right operator for the type of data that you're evaluating. For example, you could use operators such as *is greater than* or *is less than or equal to* when comparing or testing numeric values. You can use some operators, such as *is equal to* and *does not end with*, to evaluate both strings of text and numeric values.

Power Automate conditions also allow for the use of multiple conditions as well as evaluating groups of conditions. Power Automate even supports nesting conditions, as shown in *Figure 6.3*:

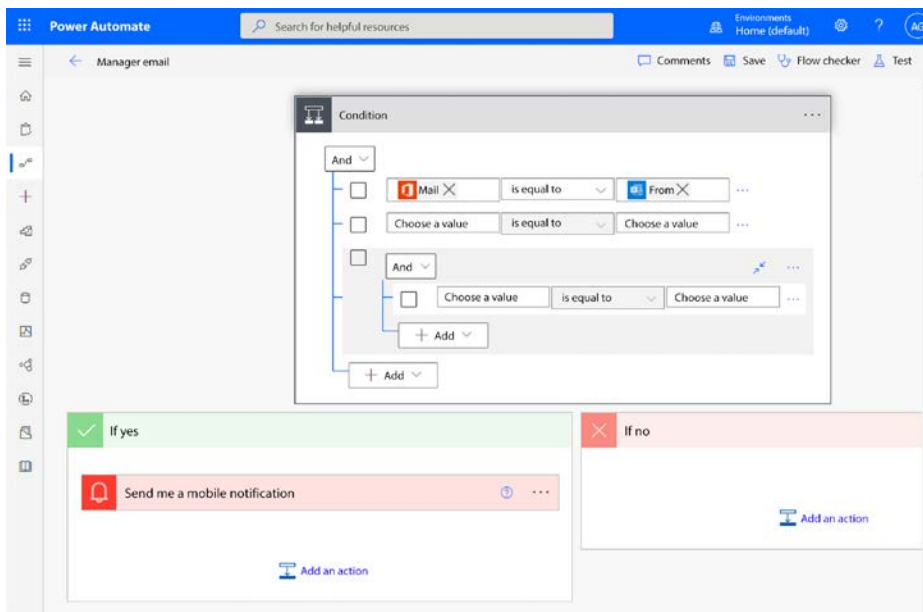


Figure 6.3: Condition groups and nesting

The result of any condition is **Boolean**: the condition evaluates to either *yes* (true) or *no* (false).

## Creating the flow

In this example, we'll use a condition to control when a notification will be sent. The goal is to send a push notification when you receive an email from your manager.



You may need to disable Experimental Features in Power Automate to get the dynamic value tokens to show up in line with the steps, as shown in the screenshots in this example. However, if you want to use advanced expressions, you can toggle Experimental Features through the **Settings** (gear) icon in the Power Automate title bar.

Follow these steps to see how the condition control is used:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **Create** and select **Automated cloud flow**.
2. Enter a name and then select the **When a new email arrives (V3)** trigger for Office 365 Outlook. Click **Create**:

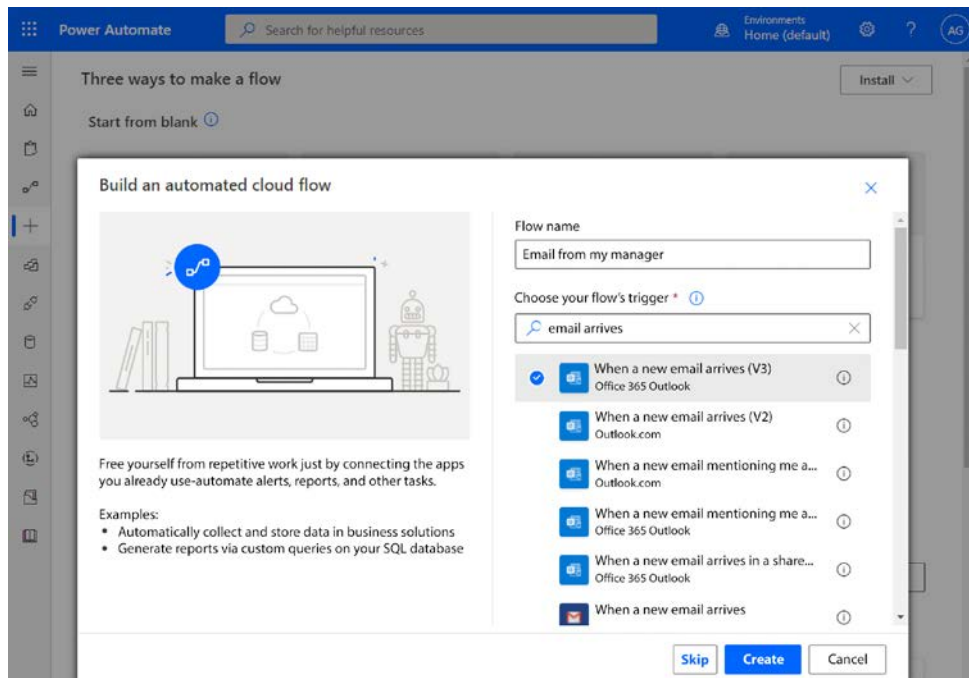


Figure 6.4: Creating a new automated cloud flow

### 3. Click **New step**:

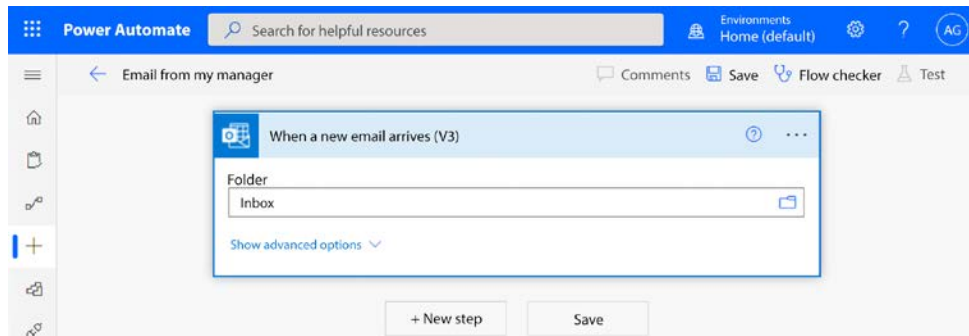


Figure 6.5: Configuring a new automated cloud flow

4. In the **Search connectors and actions** box, search for the Office 365 Users **Get my profile (V2)** action and select it.
5. Click **New step**.
6. In the **Search connectors and actions** box, search for the Office 365 Users **Get manager (V2)** action and select it.
7. In the **User (UPN)** box of the **Get manager (V2)** action, select the **User Principal Name** value under the **Get my profile (V2)** section:

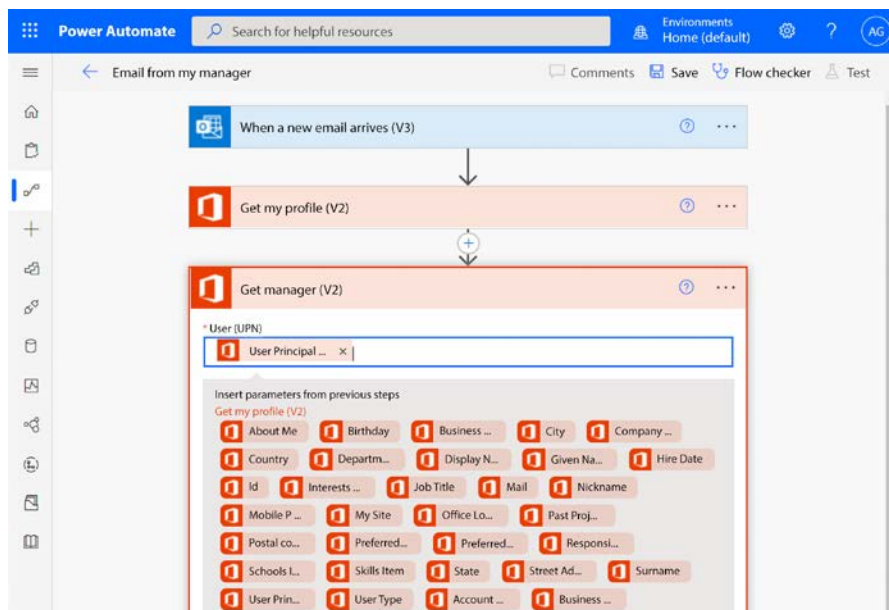


Figure 6.6: Adding the User Principal Name dynamic value token

8. Click **New step**.
9. In the **Search connectors and actions** box, search for the **Condition** action and select it:

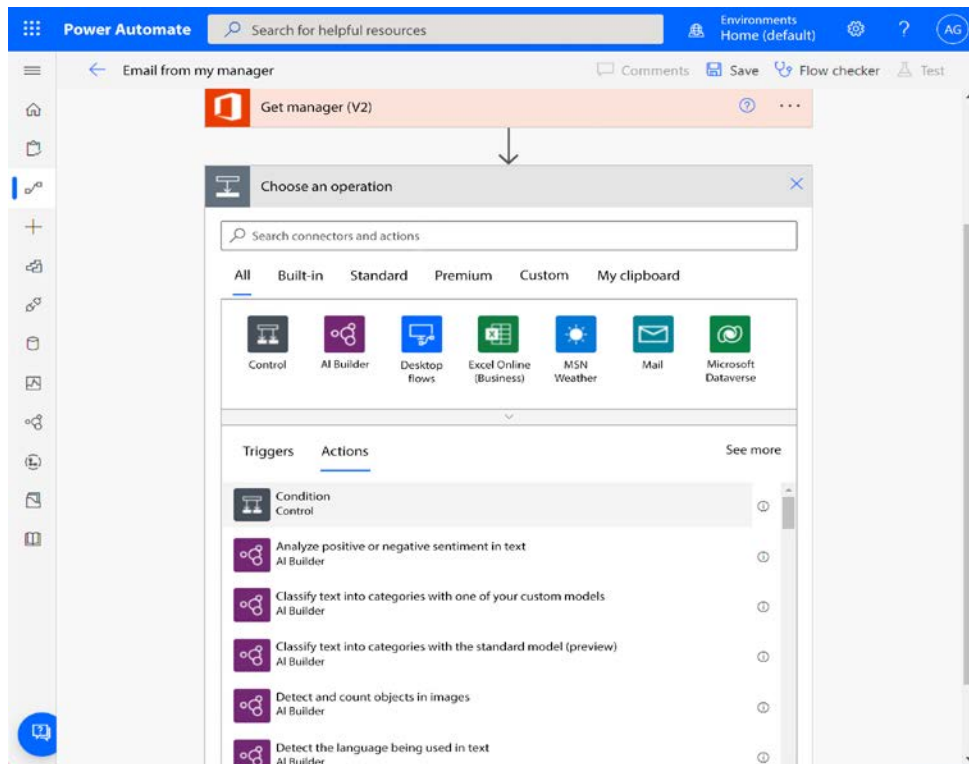


Figure 6.7: Selecting the Condition control

10. The goal of this step is to check to see whether the value of the email sender matches the result of the **Get manager (V2)** action. In the **Choose a value** box on the left-hand side of the condition, select the **Mail** value under **Get manager (V2)**:

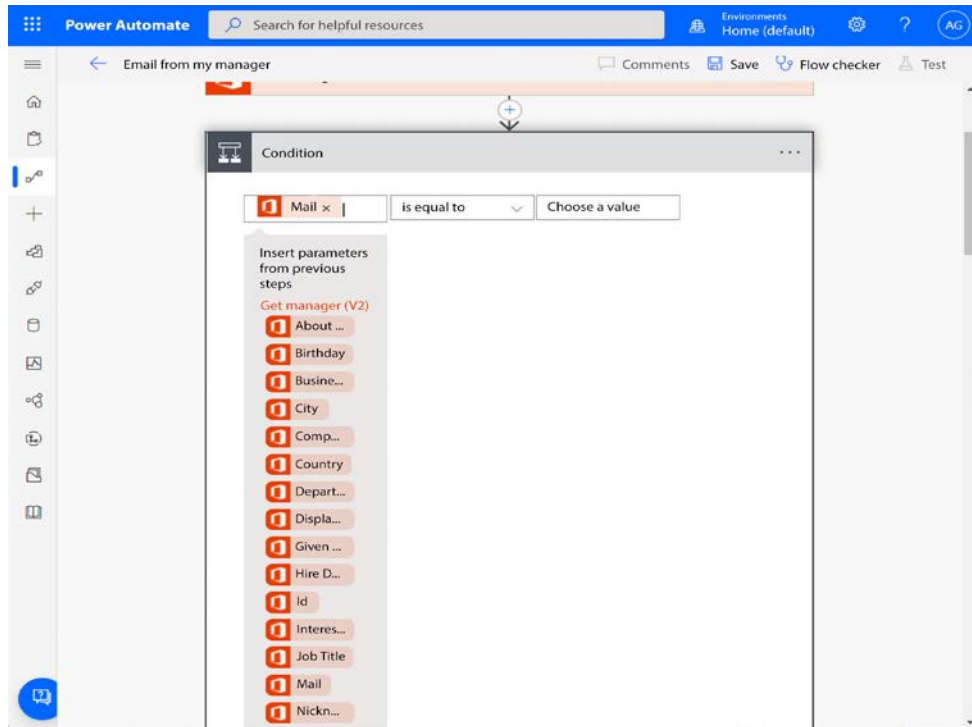
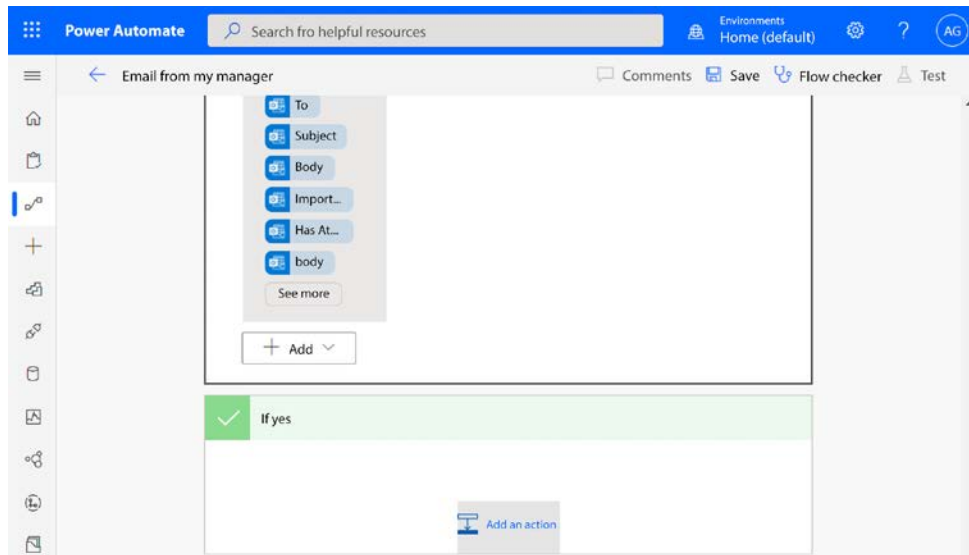


Figure 6.8: Selecting the Mail dynamic value token from the Get manager (V2) action

11. In the **Choose a value** box on the right-hand side of the condition, select the **From** dynamic content value under **When a new email arrives (V3)**.

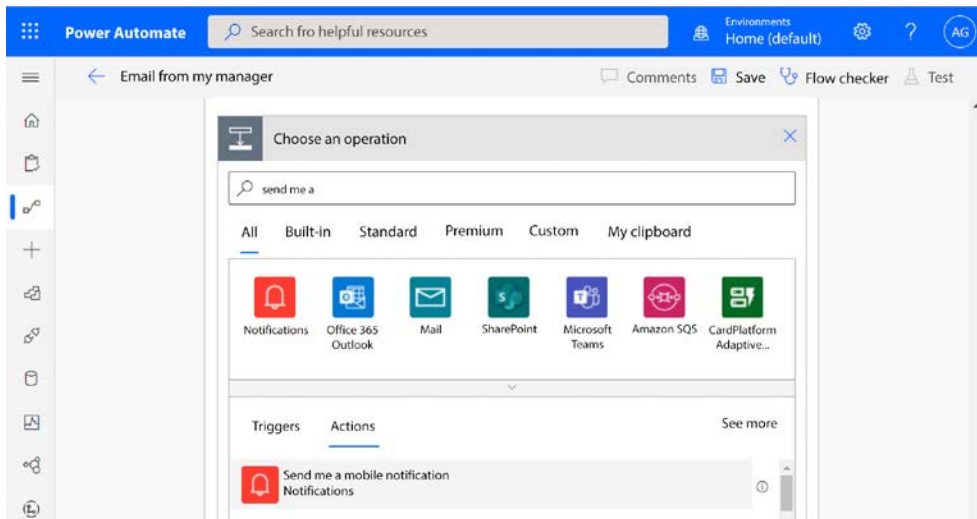


12. Under the **If yes** branch, select **Add an action**, as shown in *Figure 6.9*:




*Figure 6.9: Adding an action under the If yes branch*

13. In the **Search connectors and actions** box, select **Send me a mobile notification**:



*Figure 6.10: Selecting the action to perform for the If yes branch*

14. Customize the notification **Text** and **Link** values:



Currently, the Outlook REST API has been deprecated, and web-based deep links to messages have not been implemented in Graph API for Exchange Online. You will not be able to configure deep linking for individual Exchange Online messages.

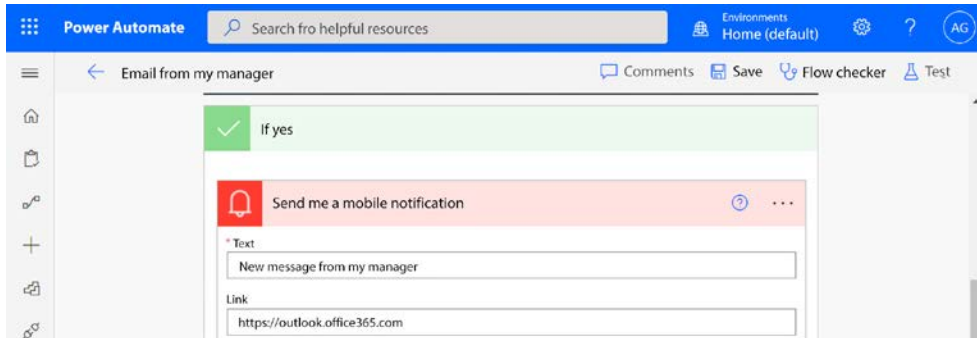


Figure 6.11: Adding the notification text and link

15. Click **Save**.

The flow has been created. You can now wait for (or ask) your manager to send you an email to trigger the notification.

## Reviewing a notification

When you receive a message that meets the conditions and triggers the notification, you should get a push notification on your device, as shown in the following screenshot:

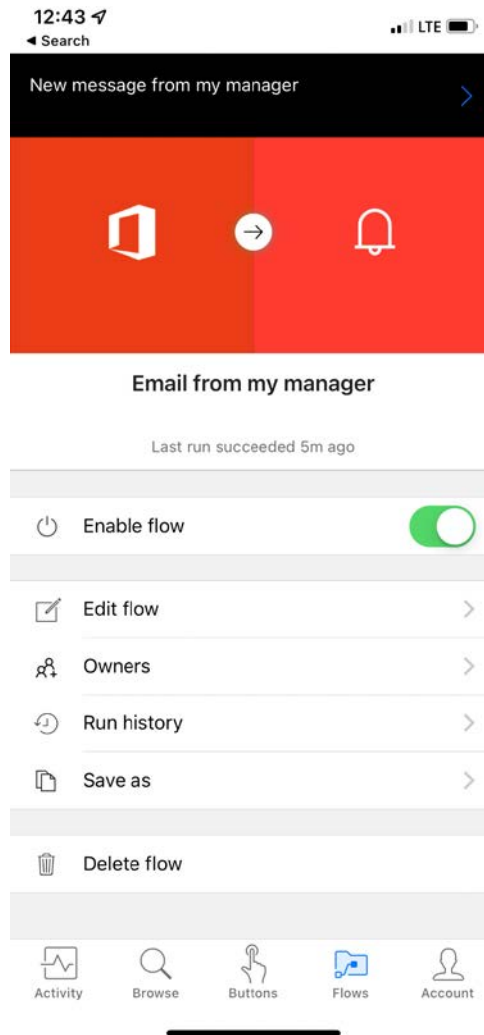


Figure 6.12: New message notification banner

You can tap on the notification banner to be taken to the Power Automate mobile app notifications page:

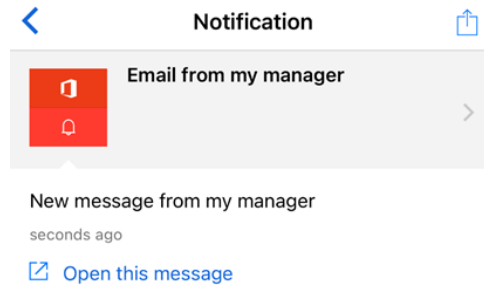


Figure 6.13: Push notification detail

From here, you can use the **Open this message** link (or whatever text you specified in the push notification link text) to open the URL specified.

## Reworking the flow

When you were configuring the flow's trigger, you may have noticed that there was a parameter that allows for filtering based on *From*. You could also choose to enter your manager's name there and perform filtering earlier instead of dynamically retrieving it with the **Get Manager (V2)** action and evaluating it using a condition action.

In the real world, your choice may depend on how portable you need your flow to be and how many runs you anticipate your flow to use. With seeded (or free) Power Automate licensing, you're limited to 10,000 actions a day. If you subscribe to a per-user Power Automate plan, your action limit is 100,000 per day. For the most up-to-date information on limits, see <https://docs.microsoft.com/en-us/power-automate/limits-and-config>.

## Summary

In this chapter, you learned about creating flows that utilize push notification actions. Push notifications can be useful for drawing a user's attention to a particular event or item that requires prompt action.

You also started learning about the power of conditions and how they can be used to enable flows to perform different actions based on the data that is received. Conditions are necessary to ensure that processing only happens when certain values are detected. Conditions are part of the Power Automate system itself and, as such, can be used with any connectors. Conditions can use expressions and dynamic content as well, just like other actions in a flow.

In the next chapter, you'll learn about sharing flows. Sharing flows will allow your team members to build on each other's work, increasing both individual and group productivity.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 7

## Working with Shared Flows

Up to this point, all of the flows that you have created or seen have been associated with your particular Office 365 user account. These types of flows are individualized and may rely on your identity to authenticate.

However, if you are part of a team or share responsibility for a work process, you may find it necessary to create flows that others can also see and manage. This is where **shared flows**, formerly known as **team flows**, can help us.

In this chapter, you'll learn the basics of shared flows as we cover the following topics:

- Understanding team flows
- Sharing a cloud flow
- Sharing a desktop flow
- Sharing a flow with run only permissions
- Managing shared flows

Team flows are important to enable business continuity in the event that one or more parties of a flow change roles or leave the organization altogether. By the end of this chapter, you'll know how to share and manage team flows.

Let's begin!

### Understanding shared flows

As mentioned in the introduction, shared flows enable multiple individuals to manage the activity and configuration of a flow. Shared flows also enable multiple individuals to view the run history and provide credentials for connectors.

Both cloud flows and desktop flows can be shared.

There are, however, some interesting features and caveats for shared flows:

- Other users, groups, or SharePoint lists can be made co-owners of a flow. The creator of the flow *cannot* be removed by another co-owner. If you are granting access to a SharePoint list, that site must be connected to the flow (such as being used to save a file).
- Co-owners of a flow can view a flow's run history.
- Co-owners of a flow can add or delete actions, conditions, or other co-owners.
- Co-owners of a flow can manage the properties of a flow (for example, whether it's enabled or disabled, its descriptions, and other general properties).
- Co-owners can delete a flow.
- Connections that are part of a flow can only be used in the context of the shared or team flow; they can't be used by other co-owners in their own flows.

Any flow can be made into a shared or team flow. No special design considerations are necessary, but if a desktop flow is shared, the user accessing the flow will need to install the Power Automate desktop app to be able to work with the flow.



Credential and connection information are part of the shared flow, so it may be possible for others to update the flow in such a way as to gain access to other information (such as using shared database credentials to execute additional queries). While others can't export the credential data itself and reuse it elsewhere, they may modify it to flow and have access to anything that the saved credential can do for that connection.

When using shared flows, you should ensure that the account and connection configurations only have access to the minimum amount of data necessary to process the flow.

Next, we'll look at converting a standard user flow into a shared flow.

## Sharing a cloud flow

Now that you've got an understanding of what additional features a shared flow provides, let's work on sharing a flow with some team members.

In this example, we're going to take the Expense Reports flow we created in *Chapter 3, Working with Email*, and update it to be a shared flow:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>) and select **My flows**.
2. Select the **Expense Reports** flow you created in *Chapter 3, Working with Email*, and then click the share icon. You can also click the ellipsis and select **Share** from the context menu:

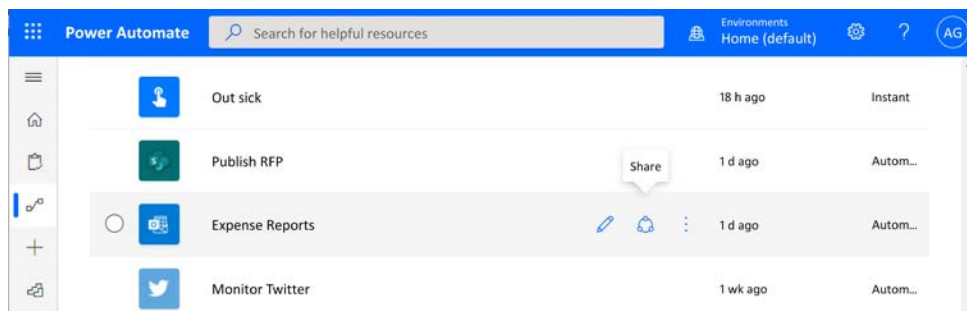


Figure 7.1: Sharing a flow

3. In the **Owners** section, under **Users and groups**, begin entering the name or address of a user or group to whom you will grant access and click to select it:

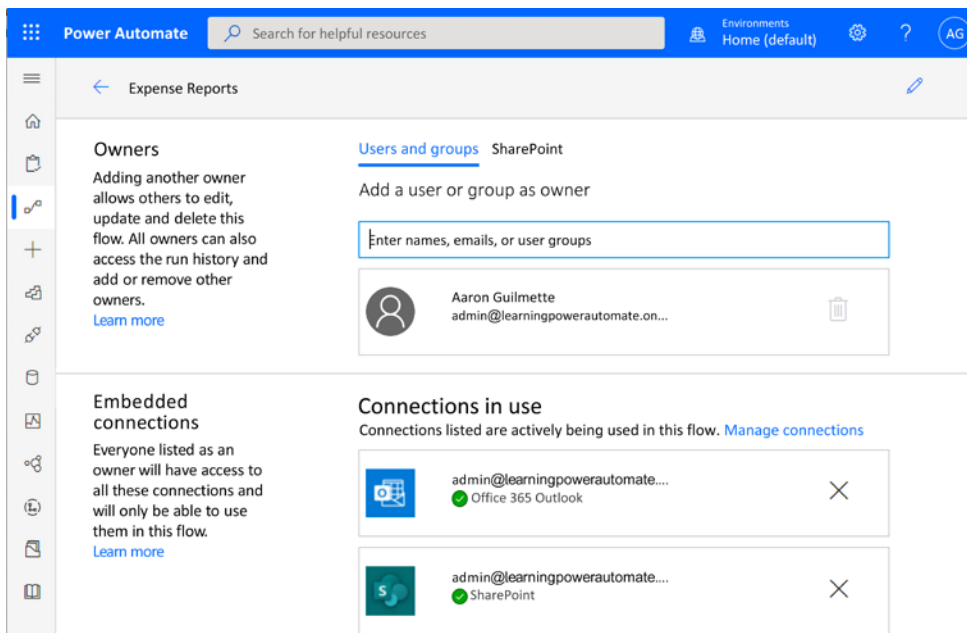


Figure 7.2: Adding co-owners to a flow



4. When adding new members, you'll be prompted to click **OK** to acknowledge what access to the connectors and data the co-owners have:

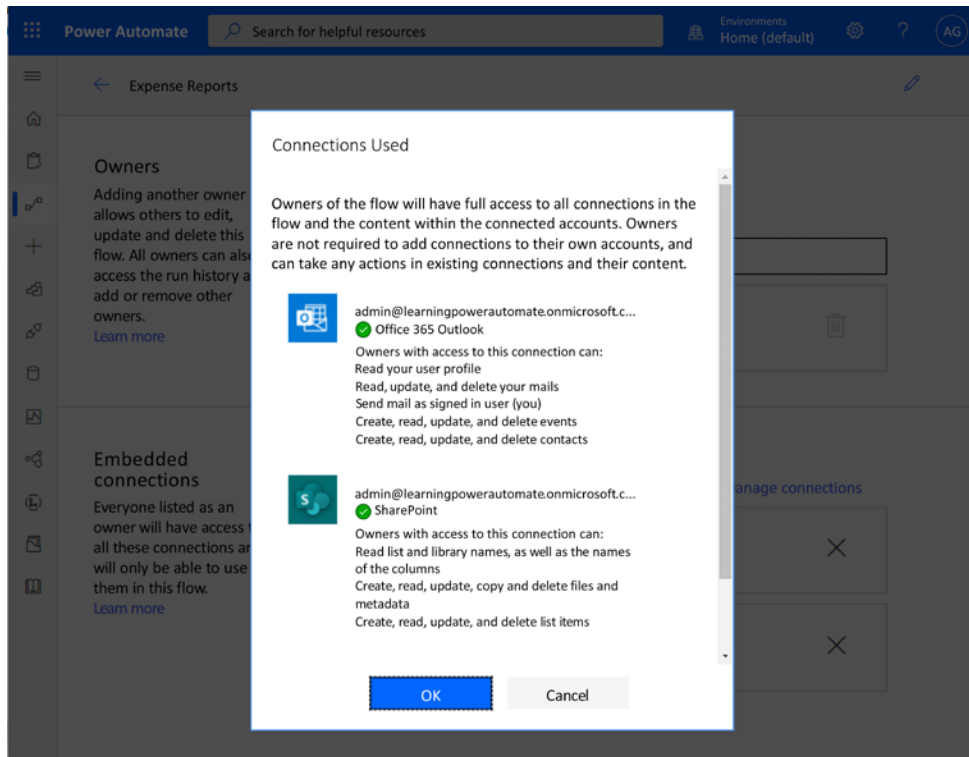


Figure 7.3: Acknowledging what permissions are conferred when sharing

5. If your flow is connected to a SharePoint site, you can grant permissions to users who have edit access to a SharePoint list or library. Select **SharePoint**, and then select the connected site and list or document library. Click **Add**:

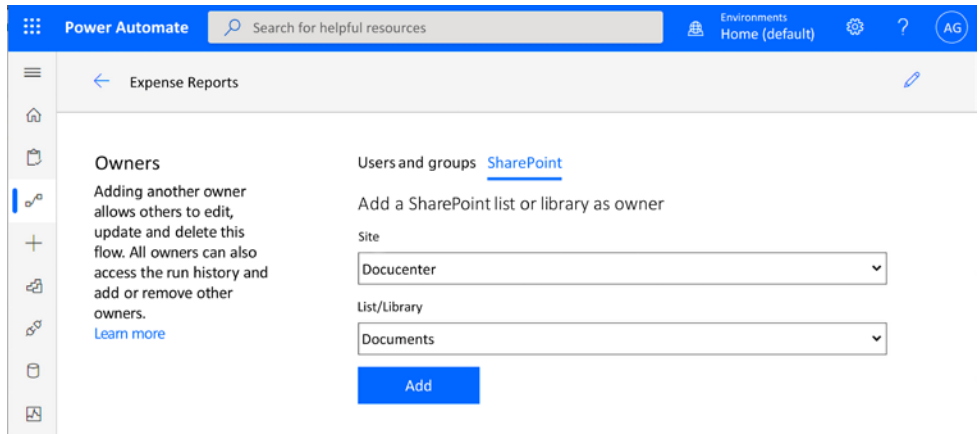


Figure 7.4: Adding a SharePoint document library as an owner

6. On the **My flows** page, the flow will be moved from the **Cloud flows** (formerly, **My flows**) tab to the **Shared with me** (formerly, **Team flows**) tab, as shown in the following screenshot:

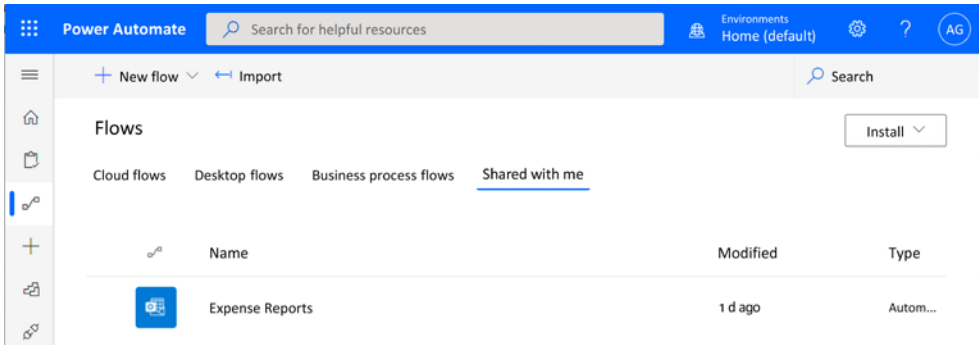


Figure 7.5: Reviewing shared flows

The flow has been shared.



At this time, even if you remove all co-owners and lists or libraries from a flow, it stays on the **Shared with me** tab.

Next, we'll look at how sharing works from within the Power Automate desktop app.

## Sharing a desktop flow

Power Automate Desktop, as mentioned in *Chapter 2, Getting Started with Power Automate*, can be used to create and manage desktop flows. At this time, there are some caveats and limitations:

- The Power Automate desktop application only shows and allows you to edit desktop flows. You cannot see cloud flows in the Power Automate desktop app.
- The Power Automate web portal shows you both cloud and desktop flows. You can edit cloud flows in the browser, but editing or running a desktop flow requires the Power Automate desktop application.
- You can share desktop flows within the Power Automate web portal interface. You cannot share desktop flows from the Power Automate desktop application.

As the product continues to mature, some of these caveats may change.

In order to complete this exercise, we'll first need to create a very basic desktop flow.

## Creating a basic desktop flow

We're going to get more into desktop flows in *Chapter 16, Introduction to Robotic Process Automation*. The flow we create in this process will just list the files in a directory but will allow you to have a flow that can be used to demonstrate how sharing a desktop flow works.

To create a desktop flow, follow this sequence of steps:

1. Launch the Power Automate desktop app. If you are running Windows 11, it's already built in. If you are running Windows 10, you will need to download it from the Power Automate **My flows** page, as shown in *Figure 7.6*:

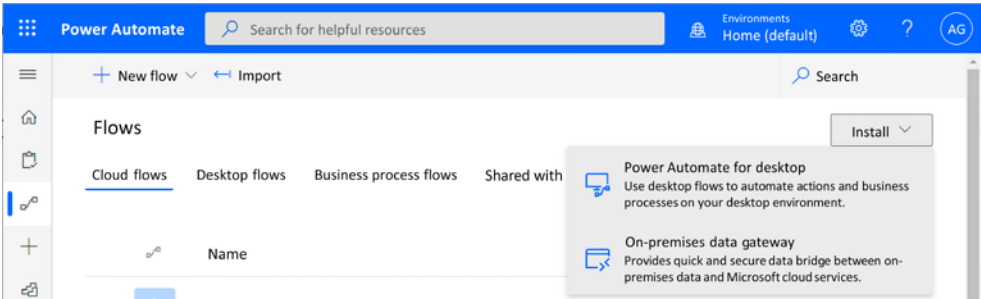


Figure 7.6: Downloading Power Automate for desktop

2. Click **+ New flow**:

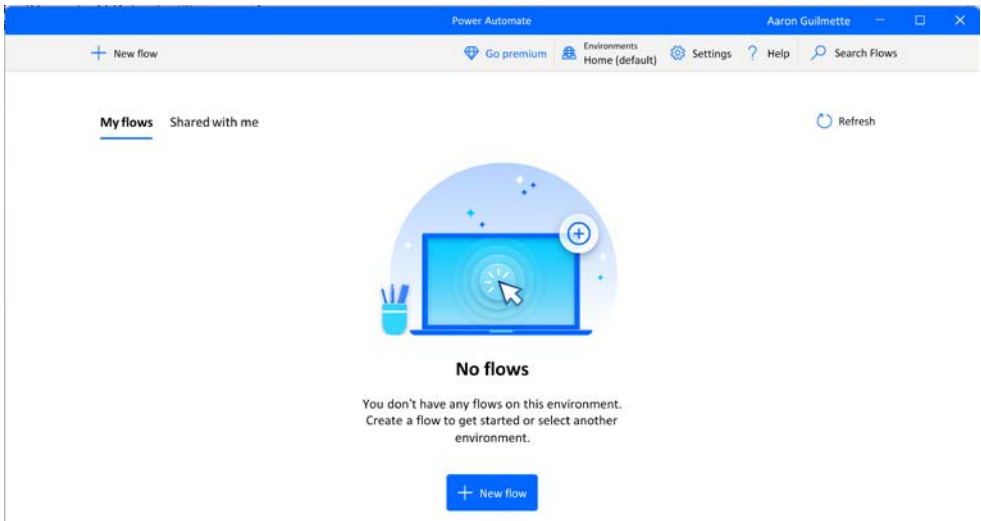


Figure 7.7: Power Automate desktop page

3. Add a name and click **Create**.

4. Once the Power Automate designer launches, expand **CMD session** under **Actions**, select **Open CMD session**, and drag it to the canvas area:

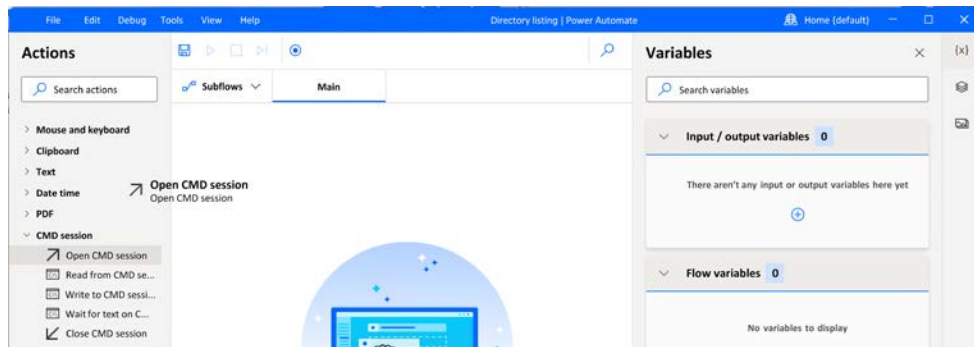


Figure 7.8: Selecting the CMD action

5. In the **Working folder** box, type `C:\` and then click **Save**:

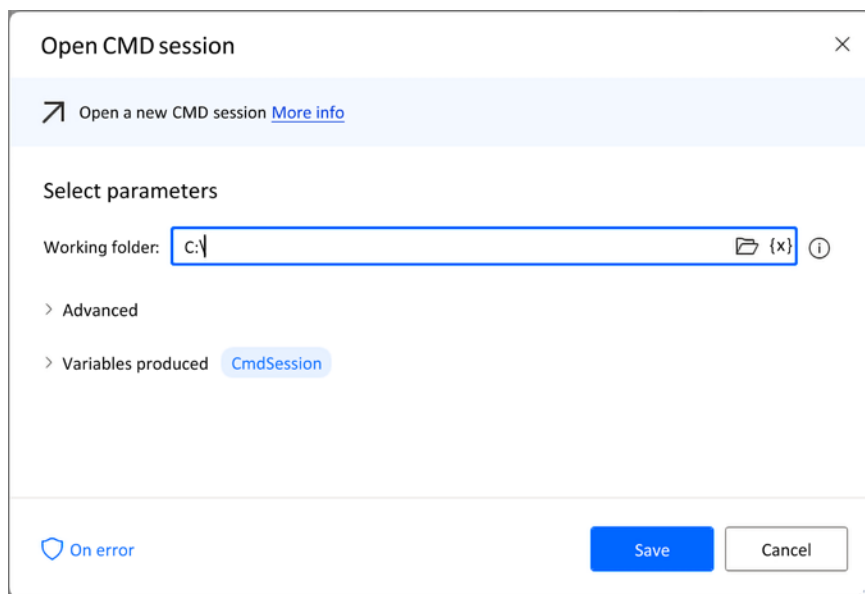


Figure 7.9: Specifying the working folder

- Next, drag the **Write to CMD session** action to the canvas area below the **Open CMD session** action:

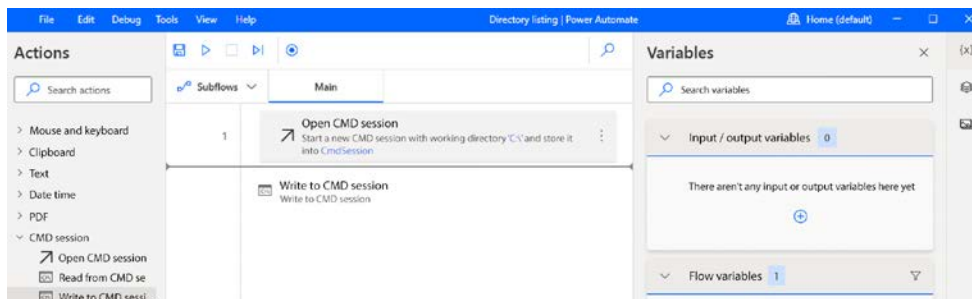


Figure 7.10: Adding the Write to CMD session action

- In the **Command** box, type `dir c:\ > dir.txt` and click **Save**:

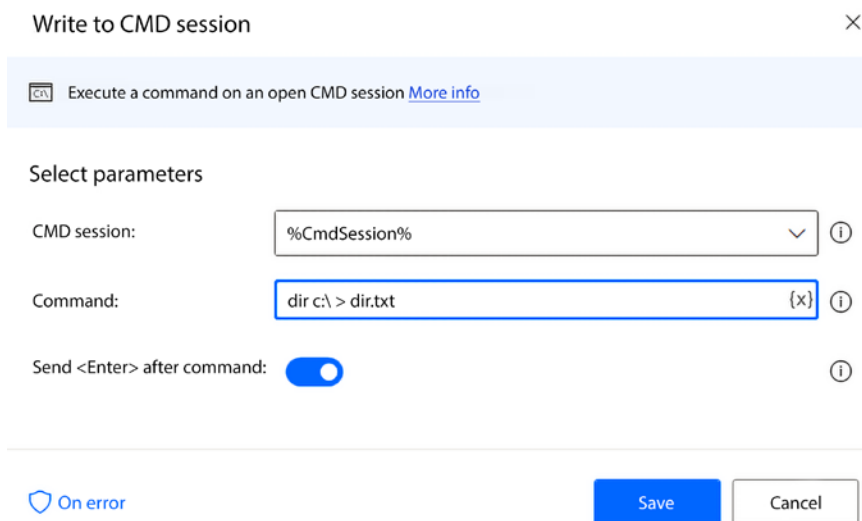


Figure 7.11: Specifying a command to execute

- Finally, drag the **Close CMD session** action to the canvas below the **Write to CMD session** action. Click **Save** in the dialog box, and then click the **Save** icon in the title bar of the Power Automate designer.

That's it! You've created a basic desktop flow. It doesn't do much, but it will be enough to show you how the sharing process works, which we'll cover in the next section.

## Sharing a desktop flow

To share a desktop flow from the Automate web portal, follow these steps:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>) and select **My flows**.
2. Select the **Desktop flows** tab:

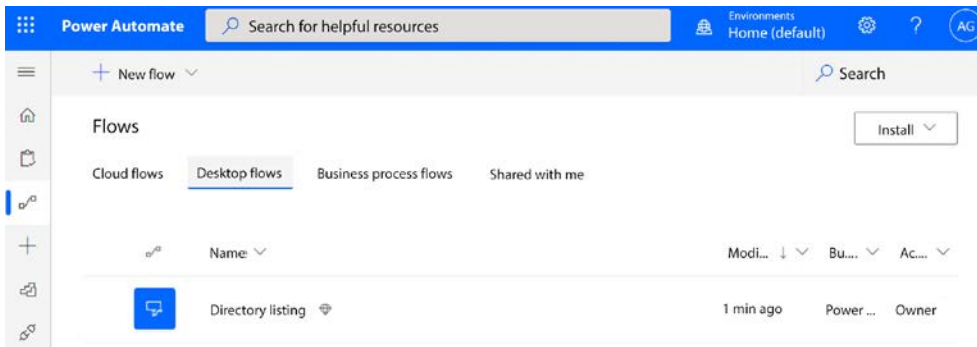


Figure 7.12: Viewing desktop flows

3. Hover over the flow, expand the ellipsis, and then select **Share**:

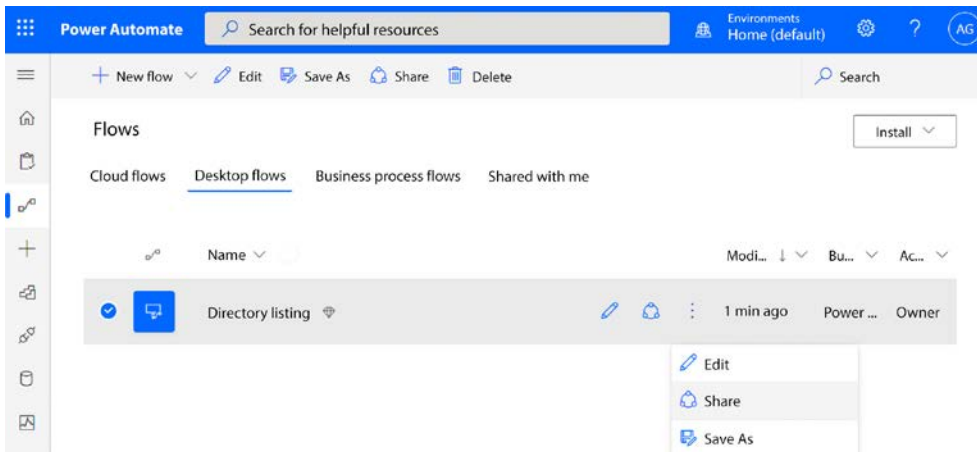


Figure 7.13: Accessing the Share activity

4. Begin typing a name or email address in the user box and select the user to add to the flow.

5. In the **Give access** box, you can select either the **User** or **Co-owner** permission:

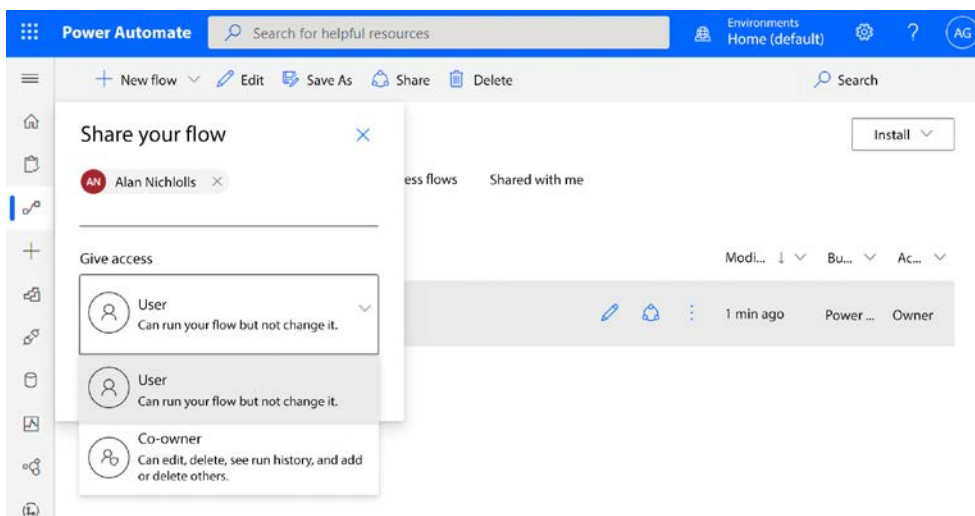


Figure 7.14: Adding a user and configuring permissions

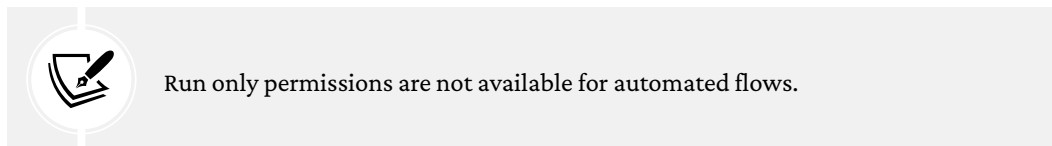
6. The added user will then be able to access the flow under the **Shared with me** tab in the Power Automate desktop application.

In some cases, you may not wish to share a flow with full co-owner permissions. While the Power Automate desktop sharing interface allows you to select the permissions level during the sharing process, the cloud flow procedure does not.

Next, we'll look at how to share a cloud flow with user or run only permissions.

## Sharing a flow with run only permissions

The **Run only permissions** feature allows you to grant a very limited set of permissions to manually triggered flows (such as button and instant flows).



With this option, users can only execute the flow. They cannot edit or modify any part of the flow and will only be granted permission to trigger it.



To grant a run only permission to a button or instant flow, follow these steps:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>) and select **My flows**.
2. Select the ellipsis for a button or instant flow, such as the **Out Sick** instant flow you created in *Chapter 5, Creating Button Flows*. Select **Details** to open the **Details** page:

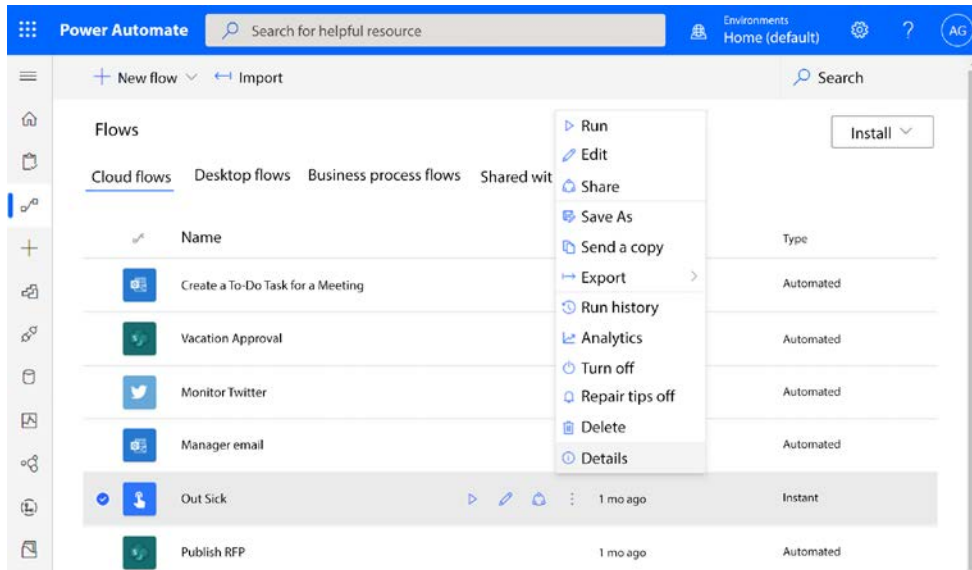


Figure 7.15: Opening the Details page

3. In the **Run only users** section, click **Edit**:

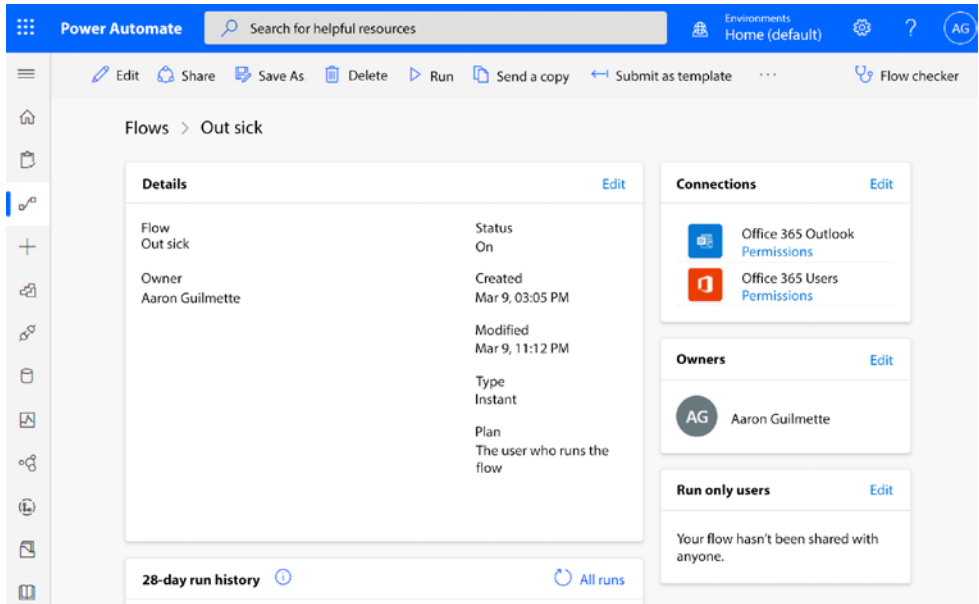


Figure 7.16: Flow details page

4. In the **Manage run-only permissions** panel, add the user or group for which you want to grant run only access. For connections that require credentials to data sources, you can select **Provided by run-only user**, for which the user will need to provide their own credentials or use those saved with the flow. The credentials provided are only in the context of this flow and not tied to any other flows a user may have:

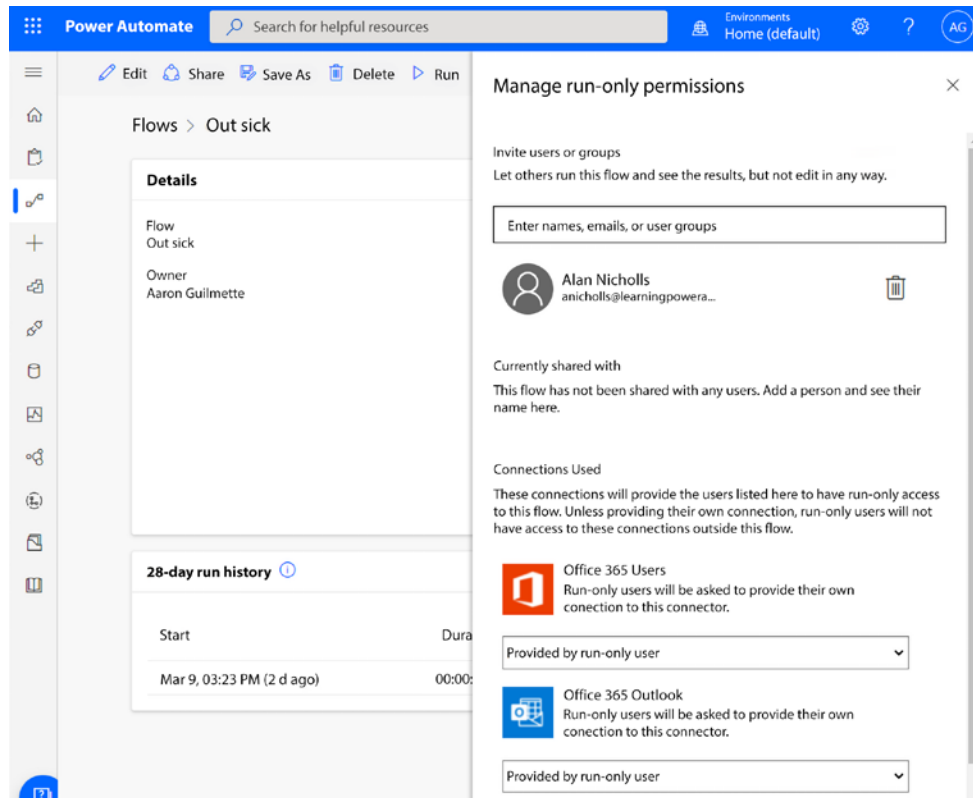


Figure 7.17: Managing run only users

5. Scroll to the bottom of the panel and click **Save**.
6. The **Details** tab will show the updated configuration with run only users.

The flow has been configured with run only permissions for the specified users and groups.



When a flow is configured with just run only permissions, it is not shown under the **Shared with me** tab. Only flows that have been configured with co-owners will be moved to the **Shared with me** tab.

Next, we'll look at some tasks related to managing shared flows.

## Managing shared flows

All co-owners of a flow can manage its various properties. In this exercise, we'll look at how a team flow looks to a sharing recipient.

Once a flow has been shared with you (or you have shared a flow with others), you can manage it like any other flow using the **My flows | Shared with me** page of the Power Automate web portal. One of the most important changes comes when the *creator* of the flow is no longer available. After the account of a creator has been removed from the organization, connections that utilize that credential will need to be updated.

In the following screenshot, notice how the status of the Office 365 Users and SharePoint connections are displayed with a red exclamation mark to denote failing credentials:

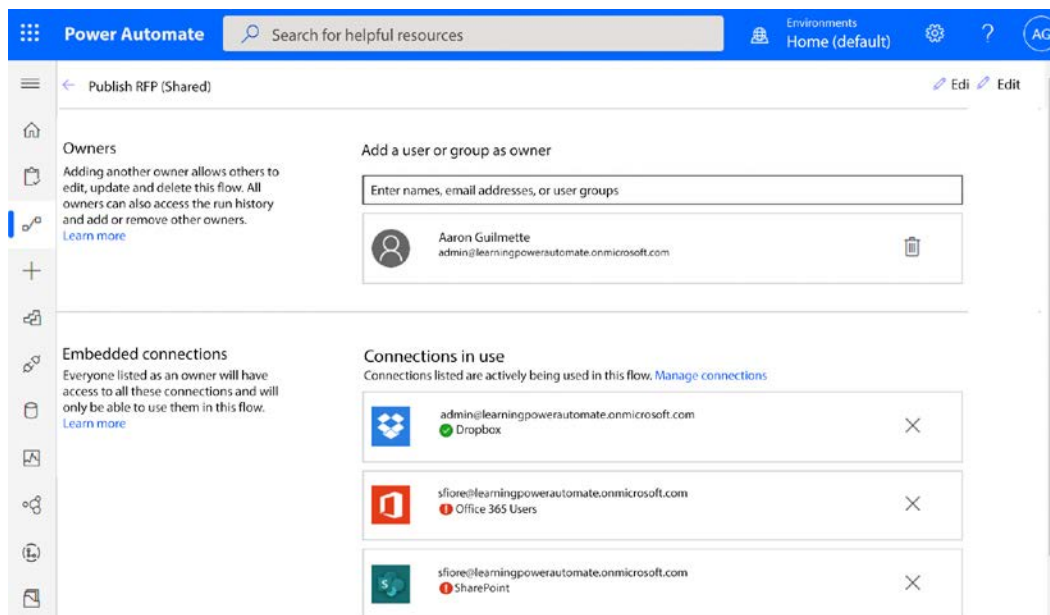


Figure 7.18: Sharing page showing invalid connectors

To resolve the situation and update the credentials, follow these steps:

1. In the **Embedded connections** section, select **Manage connections**.
2. Select the account in error.
3. Click the ellipsis, and then click **Switch account**.

4. Select a new account (or enter credentials).

You may also edit the flow to update the credential using this procedure:

1. From the **Shared with me** page, select the flow to be updated and edit it.
2. Select the object in error to expand the connection selection:

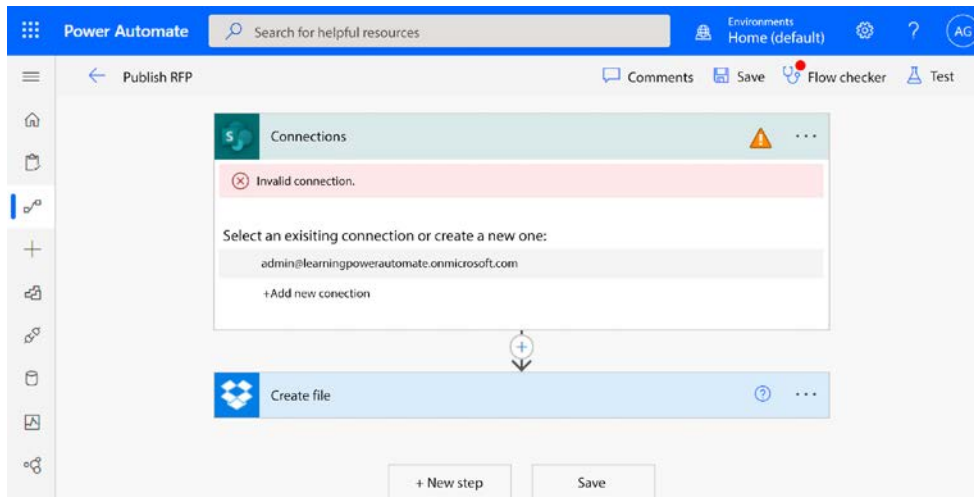


Figure 7.19: Updating the connection

3. Select a connection to use, or click **Add new connection** to create a new authenticated connection.
4. Save the flow.

You'll need to repeat this unique process whenever the account of a flow's creator is removed. This credential and authentication management process is important to successfully ensure team flows continue to work.

## Summary

In this chapter, we introduced the concept of shared flows (formerly called team flows). Shared flows allow you to share the management and ownership of a flow in your organization. Shared flow owners can manage all aspects of the flow, including adding and removing other owners and updating actions. You also learned how to manage the credentials of a shared flow after the owner's account has been removed from the Microsoft 365 organization. This process is critical to allow organizations to continue running shared flows.

In the next chapter, we're going to expand on how to use conditions in flows. Conditions will allow you to start adding both complexity and flexibility to your flows, building on the core skills you've already developed.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>





# 8

## Working with Conditions

In *Chapter 6, Generating Push Notifications*, we introduced the concept of **conditions**. Conditions can be used to evaluate parameters and data inside a flow to inform a decision. Conditions are an important part of building more complex flows.

Conditions can also be used to impose limits on the number of runs or actions that a particular flow takes – for example, making sure a certain threshold is met before committing processing resources.



In previous pricing structures of Power Automate, users were limited in the maximum number of flows they could execute per month with various plans. While the current pricing structure doesn't specifically limit the number of flow runs per month, there are still some restrictions on concurrency and long-running transactions. In addition, Azure Logic Apps, an enterprise solution using much of the same technology as Power Automate, is billed on a per-connector and per-action basis, so using conditions to limit executions from a pricing perspective can still be important. In either case, Microsoft has also imposed API limits to help ensure service availability for all users. You can review <https://docs.microsoft.com/en-us/power-automate/limits-and-config> for the most up-to-date information on limits.

Conditions, as you'll see in this chapter, can be used to expand the capability of flows or potentially combine multiple single flows into a larger, more robust flow. In this chapter, we're going to expand your understanding of conditions with the following topics:

- Understanding condition operators
- Using expressions and multiple conditions



This chapter will help you see how conditions can be used to fine-tune your flows.

Let's go!

## Understanding condition operators

Operators indicate the types of calculations used when evaluating components. You might be familiar with common mathematical operators such as the following:

- + (addition)
- - (subtraction)
- ÷ (division)
- × (multiplication)
- < (less than)
- > (greater than)

These operators are primarily used for evaluating numeric values. When manipulating objects and text in a flow, using mathematical operators may not be sufficient. While you can use numeric expressions in some places, it's also important to understand other operators that can be used to determine whether a value meets a certain condition.

Power Automate conditions can perform the evaluation of conditions using the following operators:

- **contains**
- **does not contain**
- **is equal to**
- **is not equal to**
- **is greater than**
- **is greater than or equal to**
- **is less than**
- **is less than or equal to**
- **starts with**
- **does not start with**
- **ends with**
- **does not end with**

Some of these can function with numerals (such as **is greater than or equal to**), while some work with alphabetical characters (such as **starts with**).



It is possible to use *is* comparison operators such as *is greater than or equal to* with strings and text, but it may leave you with unexpected results or make your flow harder to troubleshoot. The results in using numeric comparison operators will be based on the alphabetic position of the string as opposed to the actual values. For example, evaluating *abc is greater than def* will result in *False*, since the first character of the string *abc* has a lower alphabetic position value (1) than the first character of the string *def* (4).

Reviewing the flow you created in *Chapter 6, Generating Push Notifications*, you can see how conditions can be used to compare dynamic content values:

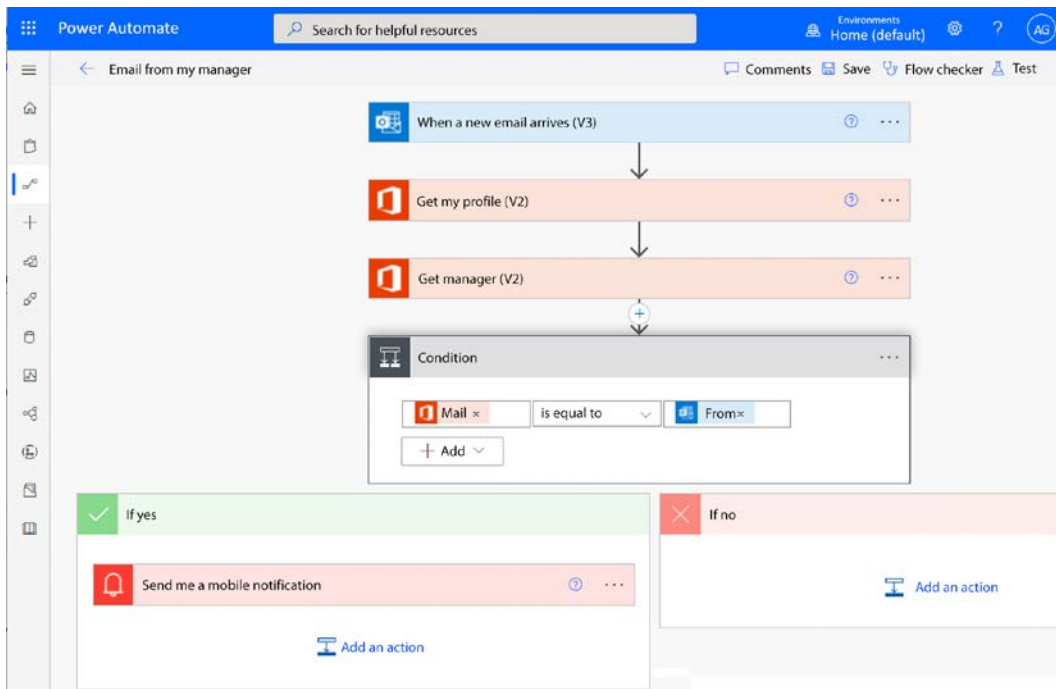


Figure 8.1: Evaluating conditions

In this case, we didn't want to send a mobile notification *every* time we received an email; we only wanted to generate a notification if the user's manager value was the same as the email sender's, effectively putting a condition or filter on what would generate a notification. This was accomplished with the **is equal to** condition, which can be used to evaluate strings or numerals.

In the following example (based on the *Monitor Twitter* flow from *Chapter 2, Getting Started with Power Automate*), you can see that we have modified it to include a condition. The updated flow includes a condition that evaluates whether the account that posted the tweet has at least 100 followers, and if it has, it then initiates a post to a Teams channel:

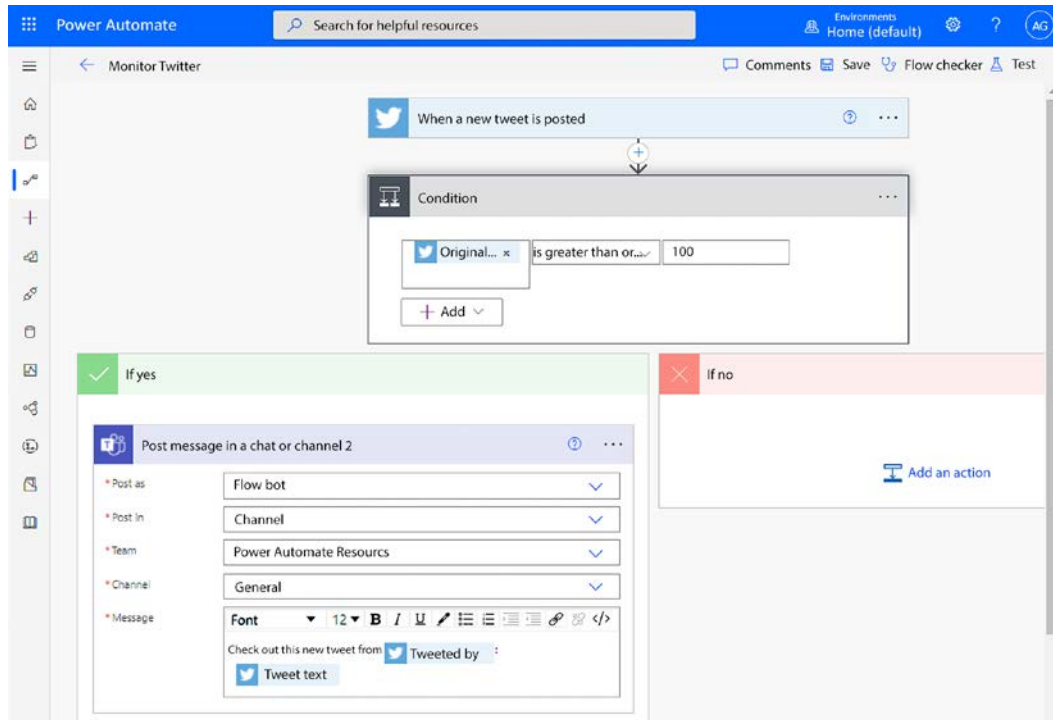


Figure 8.2: Implementing a condition to count Twitter followers

As you can see, evaluating text or numeric strings is relatively straightforward with a condition.

## Evaluating objects

In addition to being able to evaluate text strings, numbers, the results of expressions, or values from dynamic content tokens, Power Automate Desktop also has special conditional actions to deal with different types of objects:

- *If file exists*: Checks whether a file exists or not before executing a block of actions
- *If folder exists*: Checks whether a folder exists or not before executing a block of actions
- *If window*: Executes a block of actions if a specific window is open
- *If window contains*: Executes a block of actions if a specific text or UI element exists in a window

- *If image*: Executes a block of actions if a specific image is found on the screen
- *If web page contains*: Executes a block of actions if a web page contains a specific element or text value

Next, we'll look at using an expression as part of a condition's equation.

## Using expressions and multiple conditions

Perhaps evaluating conditions based on a single static value (such as the sender of an email message) doesn't provide the granularity, flexibility, or portability that a flow requires. In that case, you can also use multiple dynamic content values in a condition.

You've already seen examples using expressions as part of dynamic content, such as using `formatDateTime()` and `utcNow()` in *Chapter 4, Copying Files*. In that example, expressions were used to help generate the value for the folder name to store an expense report. You can also use expressions or functions as part of a condition.

Conditions that perform more than one evaluation are commonly referred to as **advanced conditions**. As you will see in the following instructions, you can simply use the **Add** button and continue adding criteria. When adding multiple criteria to evaluate, there are two operators available: **AND** and **OR**. Selecting **AND** will require all conditions to evaluate as `true` in order for the control to evaluate as `true`. Selecting **OR** will only require one of the conditions to evaluate as `true` in order for the whole condition control to evaluate as `true`.

In the following sections, we'll look at two scenarios for creating an advanced condition utilizing multiple criteria. The first will use multiple conditions together, and the second will use condition groups.

### Adding multiple conditions

Now that you understand what kinds of things conditions can be used to evaluate, let's look at integrating some conditional logic to add complexity to a flow. Changing the behavior of a flow using conditions is an important concept that will allow you to become very precise in handling business scenarios.

The following example shows adding a condition to our **Monitor Twitter** flow to evaluate whether a tweet's language is in either English or Spanish:

1. From the Power Automate portal, select **My flows** and then select the **Monitor Twitter** flow for editing.

- Click the + icon between the **When a new tweet is posted** trigger and the **Post message in a chat or channel** action to insert a new step, as shown in *Figure 8.3*:

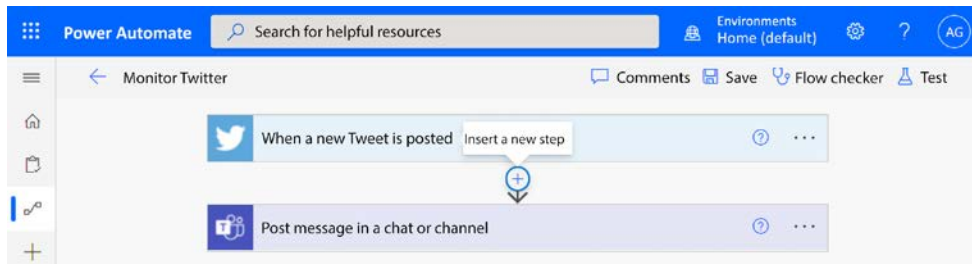


Figure 8.3: Adding a new step between existing steps

- Select **Add an action**.
- In the **Choose an operation** box, search for and add the **Condition** control:

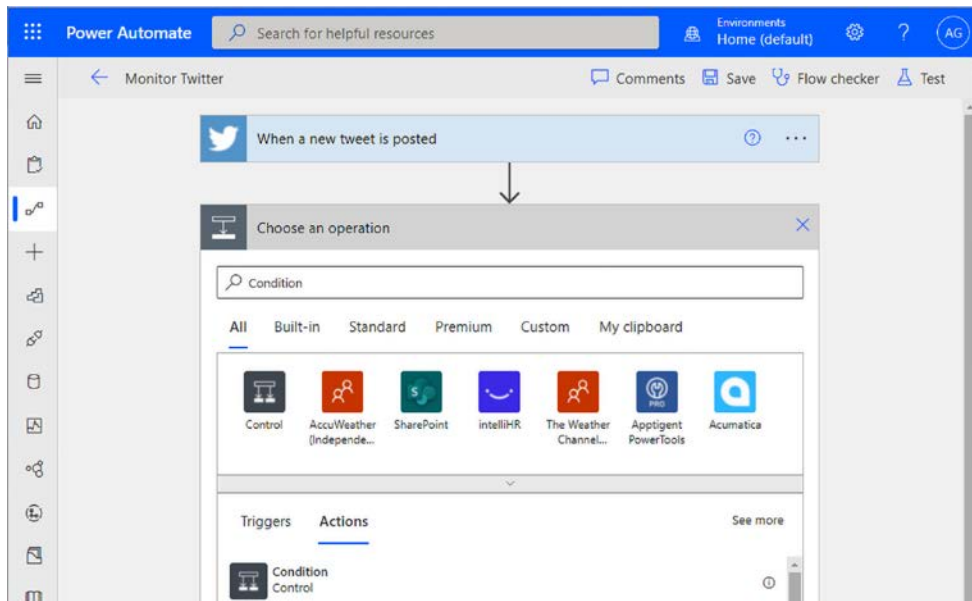


Figure 8.4: Adding a condition

- From the **Dynamic content** menu, select the **Tweet language** token and the **is equal to** operator, and then enter **en** as the value. This adds English as a condition in order for the control to evaluate as *true*.

6. Select the **Add** icon, and then click **Add row**:

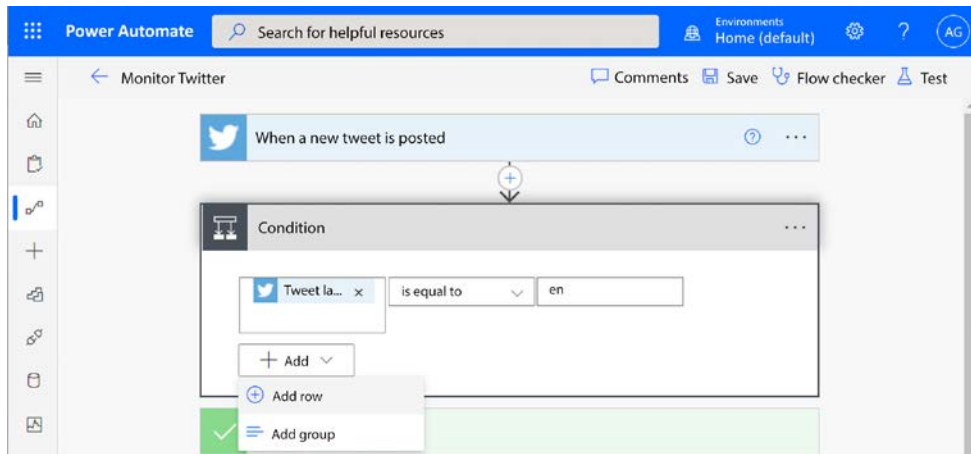


Figure 8.5: Adding a row to the condition

7. Select the **Tweet language** dynamic content token and the **is equal to** operator, and then enter **es** as the value (or another two-digit language code). This adds Spanish as a condition in order for the control to evaluate as true. At this point, the Tweet language must evaluate to *both* English and Spanish in order for the condition to be true. While this is a valid configuration from a syntax perspective, it will result in no content being able to pass the condition.
8. Select the drop-down box under the **Condition** header and then select the **Or** operator:

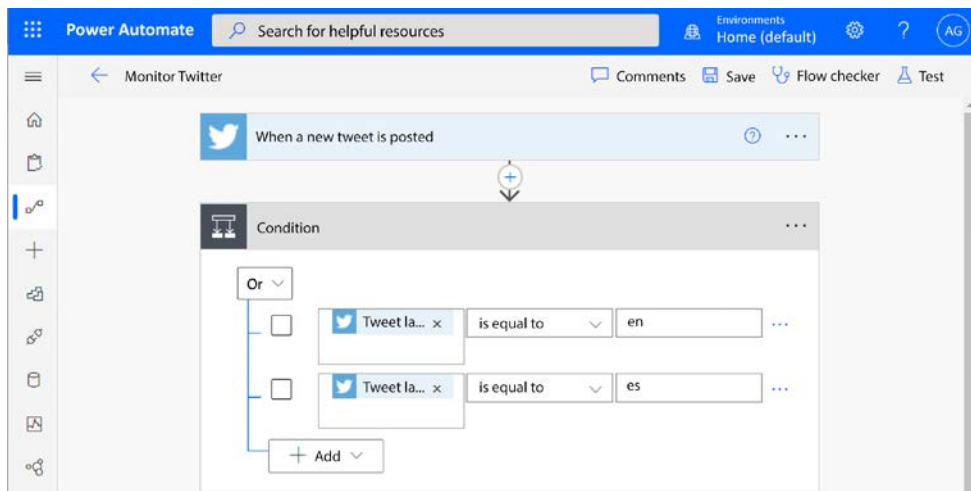


Figure 8.6: Updating the operator to OR

9. Click the **Post message in a chat or channel** action and drag it to the body of the **If yes** branch box, as shown in *Figure 8.7*:

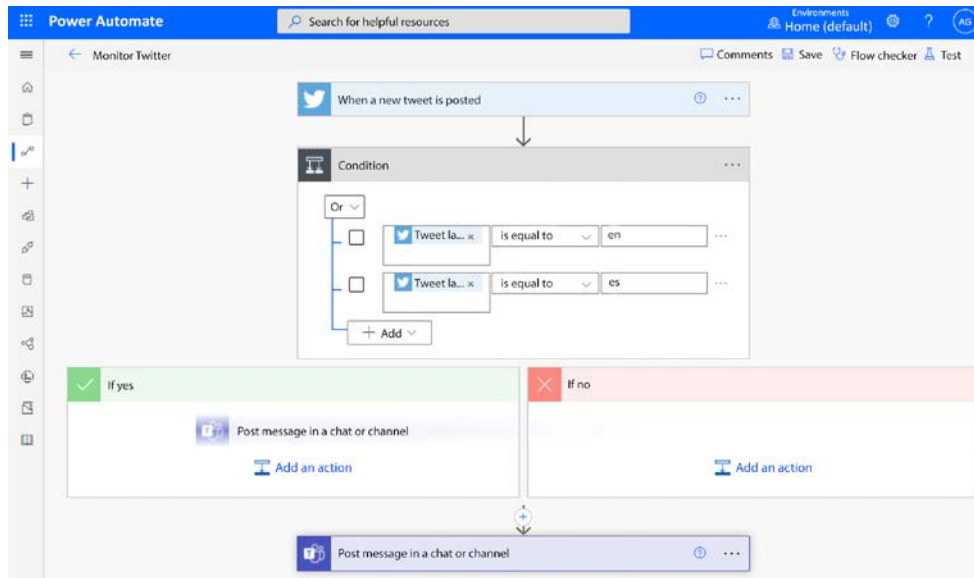


Figure 8.7: Moving the action into a condition branch

10. Click **Save**.

At this point, a tweet will need to be posted where the language is either English or Spanish in order for the condition to evaluate as true. This simple example shows how you can use conditions to help filter or identify content matching specific business criteria and act upon it.

In the next section, we'll introduce the concept of condition groups and how they can be used to further control the processing of a flow.

## Adding condition groups

In the previous example, we configured a condition where only one criterion needed to be met in order to process the flow. However, there may be scenarios where you need to select multiple values from one or more *groups* of conditions. Groups allow you to specify one or more conditions from one or more groups that need to be met in order to satisfy the control.

In this example, we'll reconfigure the flow to meet one of each of the following conditions:

- The tweet language is either English or Spanish
- The account tweeting must have accumulated more than 100 followers or favorited more than 100 items

To accomplish this, we'll create two logical groups: one that will evaluate the languages and one that will evaluate the follower and favorite counts.

Follow these steps to see how to create condition groups:

1. From the Power Automate portal, select **My flows** and then select the **Monitor Twitter** flow for editing.
2. Select the **Condition** control that you created in the previous exercise to expand it.
3. Select the checkboxes for the two Tweet language conditions, click the ellipsis in one of the rows, and then select **Make group**:

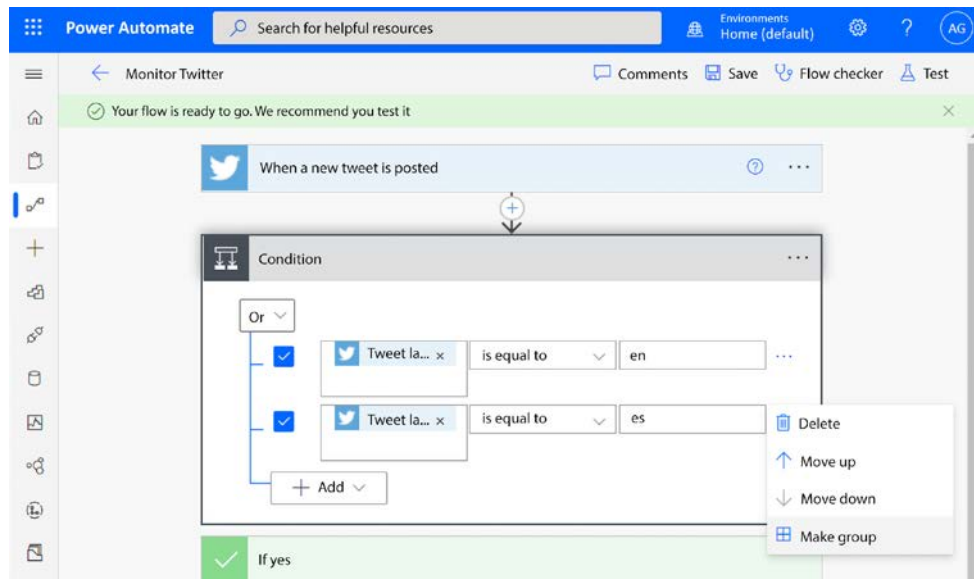
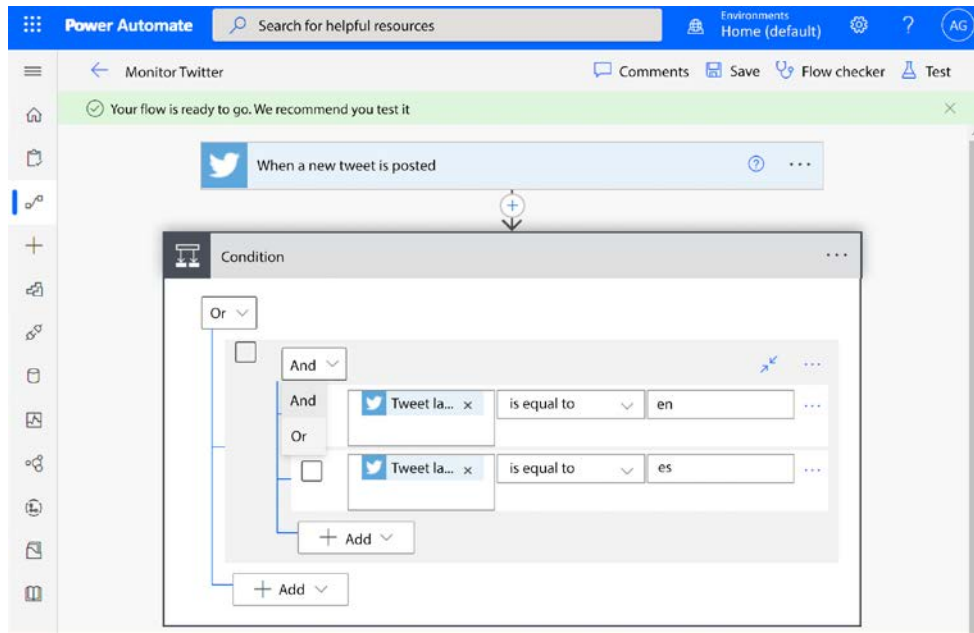


Figure 8.8: Moving existing conditions into a condition group



4. By default, Power Automate configures the operator for the new condition group as **AND**. Select the drop-down and change it to **OR**, as shown in *Figure 8.9*:



*Figure 8.9: Updating the condition group operator*

5. Click the **Add** button at the bottom of the **Condition** control and select **Add group**, as shown in *Figure 8.10*:

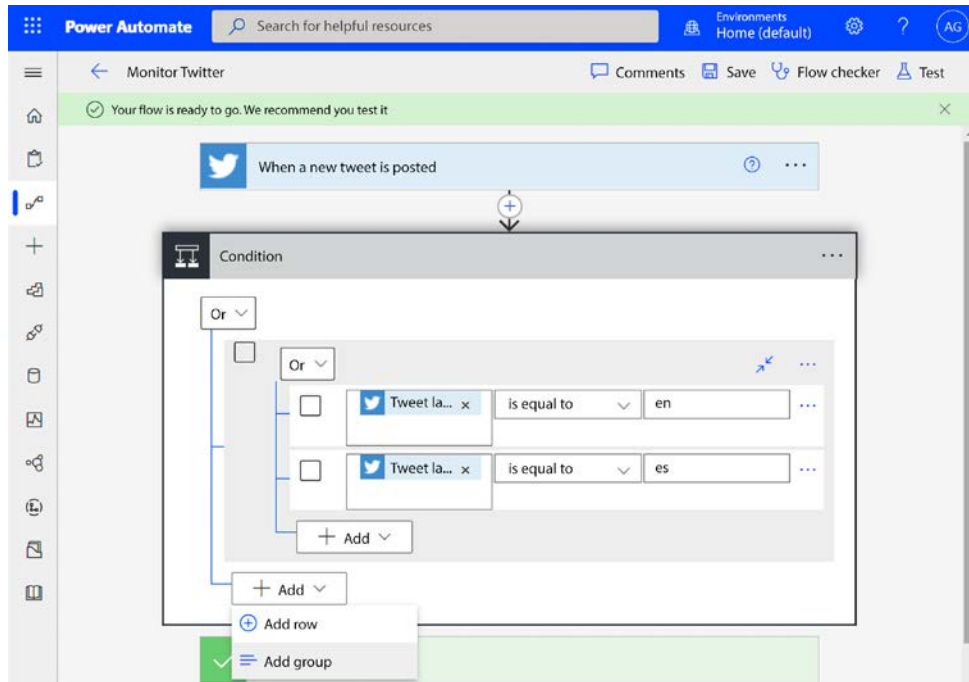


Figure 8.10: Adding a new condition group

6. In the new condition group, change the operator to **OR**.
7. In the new condition group, select the first text box in the row. Add the **Followers count** dynamic content value, select the **is greater than or equal to** operator, and then enter a value of 100.

8. In the new condition group, click the **Add** button and select **Add row**. Select the **Favourites Count** dynamic content value, select the **is greater than or equal to** operator, and then enter a value of 100.
9. Change the top-level operator for the overall condition from **OR** to **AND**. See *Figure 8.11* for the completed condition control:

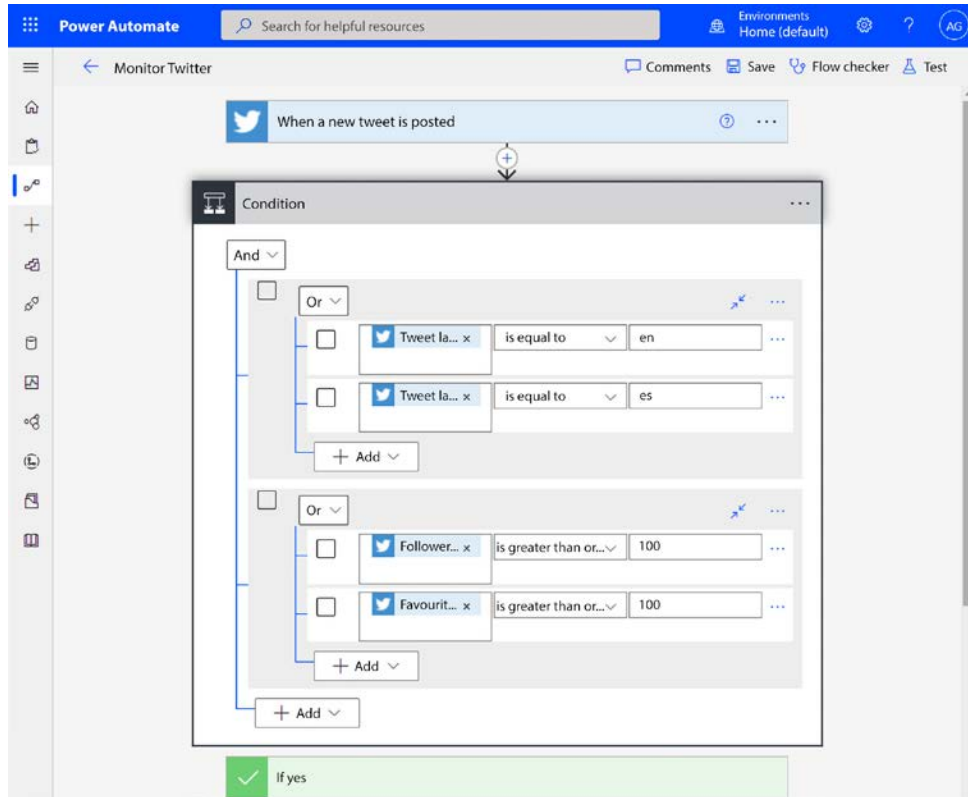


Figure 8.11: Reviewing the completed condition control

10. Click **Save**.

11. Using another browser window, prepare a tweet that meets the criteria to trigger the flow. If you don't have access to a Twitter account that meets the required number of followers or favorites, you can update the conditions to align with an account that you'll send a tweet from.
12. In the Power Automate window, click **Test**, select the **I'll perform the trigger action** radio button, and then click **Test**.
13. Switch back to the previous browser window and send the tweet.
14. Return to the Power Automate window and verify the results by expanding both the trigger and conditions. If you met the conditions to post to the Teams channel, you can go look there as well to see your tweet!

As you can see from the previous screenshot, at least one of each condition group was successfully met and the condition evaluated as *true*, causing the *Yes* branch of the condition to execute.

## Using a switch condition

In most cases, you'll probably use conditions to check for **Boolean** (*true/false*) values. However, there are some instances where a flow needs to be able to evaluate more than just *yes/no*, *equal/not equal*, or *greater than/less than*. While you could accommodate those more complex evaluations with nested conditions, Power Automate provides another construct: **switch**.

A switch condition allows Power Automate to choose from more than one branch or path. If you're familiar with scripting or development languages, the function and syntax of a switch or case statement will be familiar.

A switch condition provides an opportunity for the flow to match from a set of values; if no value matches, there is also a default action or path that the flow can take.

Consider the previous flow where you added a condition to check the language of a tweet. Instead of posting a notification of the messages to the Teams channel in English, what if you wanted to post a notification using different languages, based on the language of the tweet?

Let's edit that flow and make a few changes:

1. Open the **Monitor Twitter** flow from the previous exercises.

2. Expand the **If yes** branch:

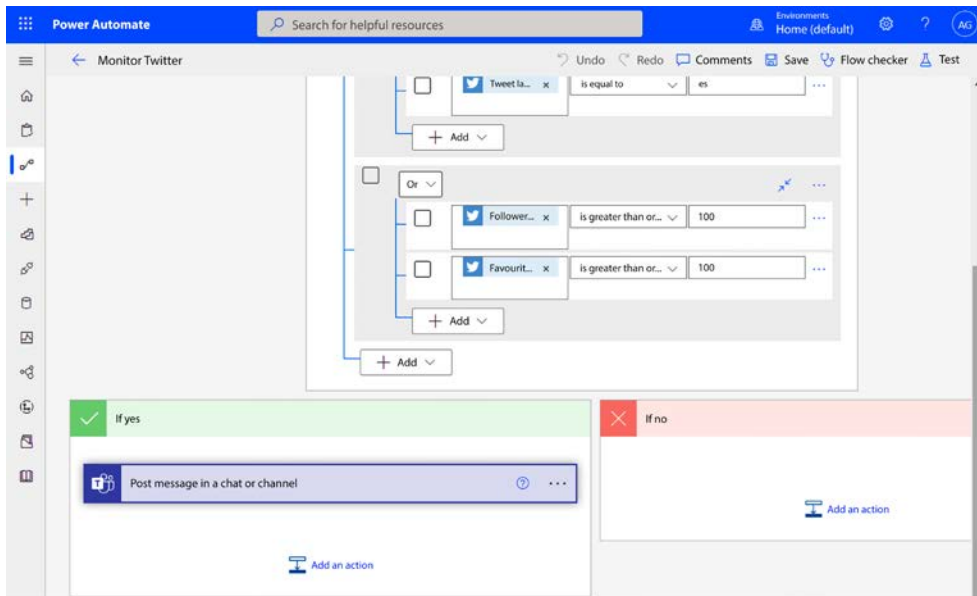


Figure 8.12: Editing the Monitor Twitter flow

3. Click **Add an action**.
4. Choose the **Switch** control action, as shown in Figure 8.13:

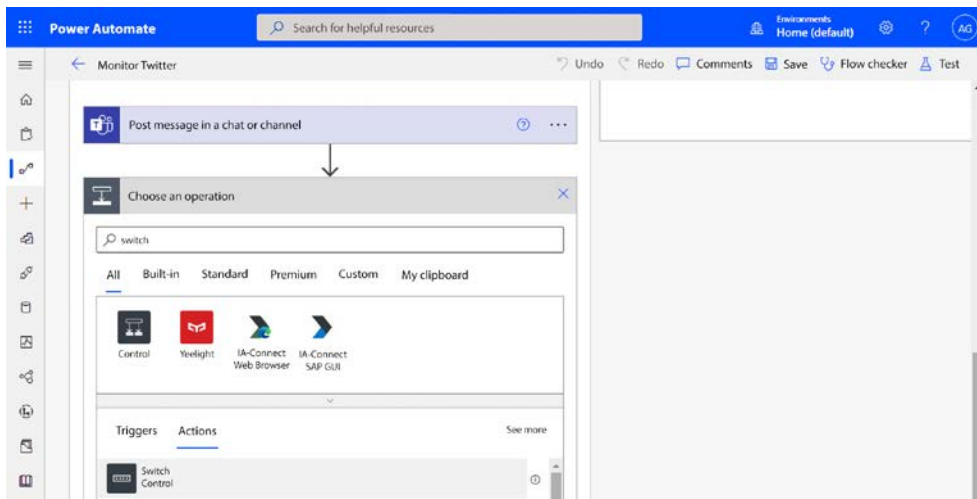


Figure 8.13: Adding a Switch control

5. In the **On** field of the **Switch** action, add the **Tweet language** dynamic content token. This indicates that we are going to perform different actions based on the evaluation of the tweet's language code:

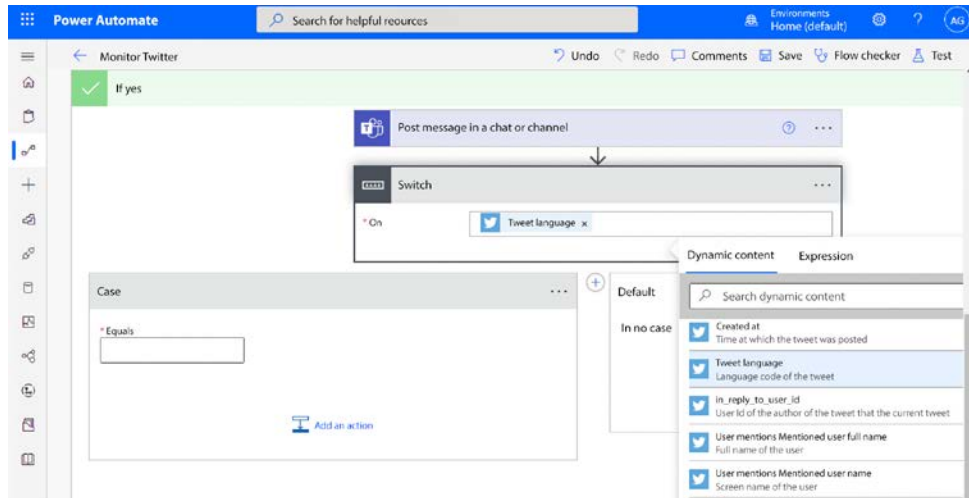


Figure 8.14: Adding dynamic content

6. In the **Equals** field of the **Case**, add the text **en**. This is the action that will be taken if the tweet's language code is English:

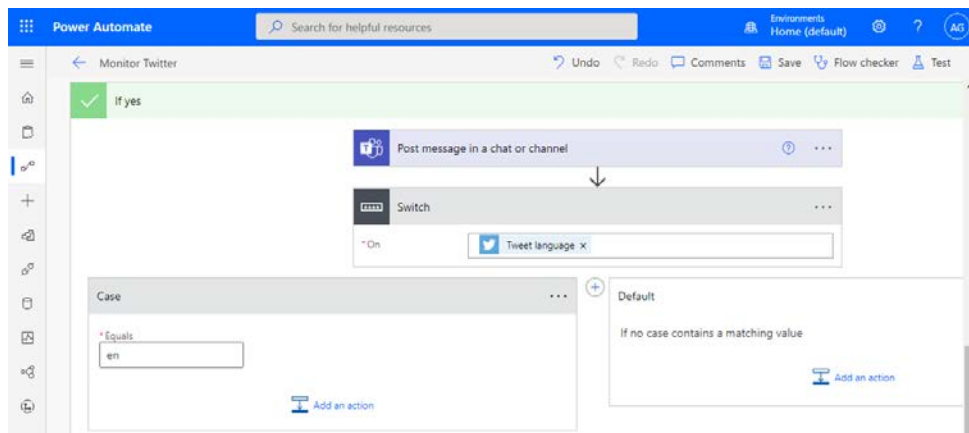


Figure 8.15: Setting the English language value code

7. Next, drag the **Post message in a chat or channel** action into the body of the **Case** action:

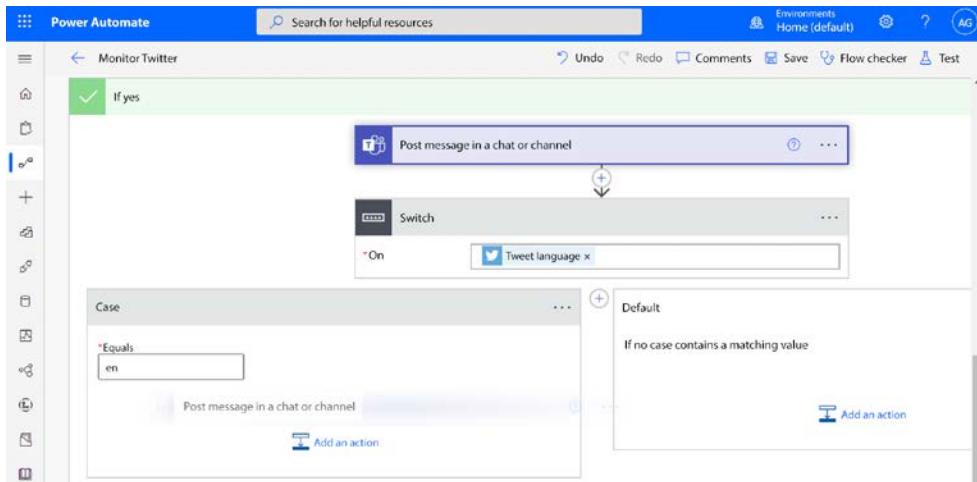


Figure 8.16: Rearranging the actions

8. Next, click the + icon between the **Case** action and the **Default** action to add a new case condition, as shown in Figure 8.17:

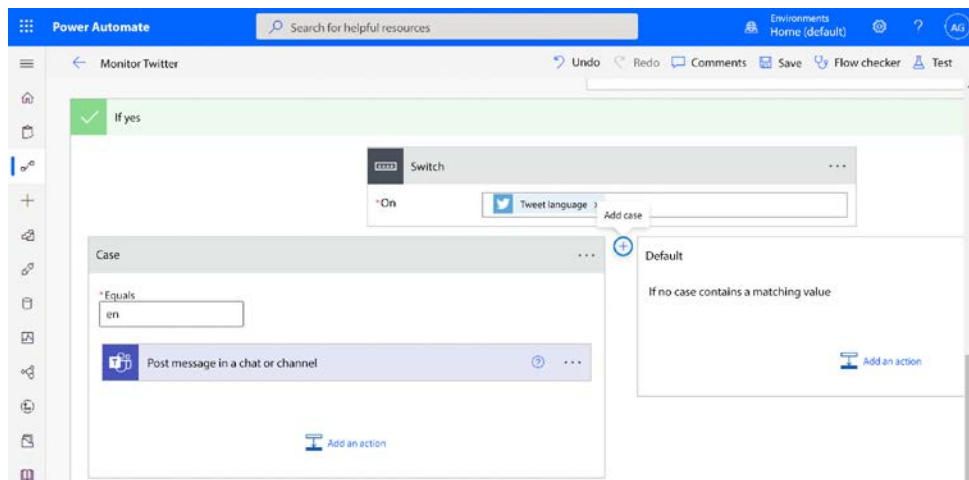


Figure 8.17: Adding a new case

9. By default, the new case will be named **Case 2**. In the **Equals** field for the **Case 2** action, enter the text **es**. We'll use this case condition if the tweet's language code maps to Spanish:

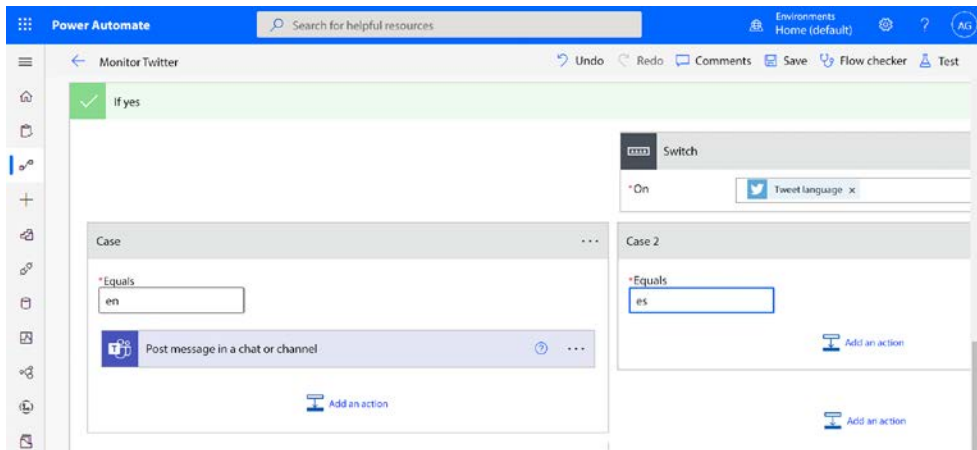


Figure 8.18: Specifying a condition for the new case

10. Inside the **Case 2** action, click **Add an action**.
11. Configure the action as you did in the previous exercise, selecting **Flow bot** as the **Post as** value and **Channel** as the **Post in** value:

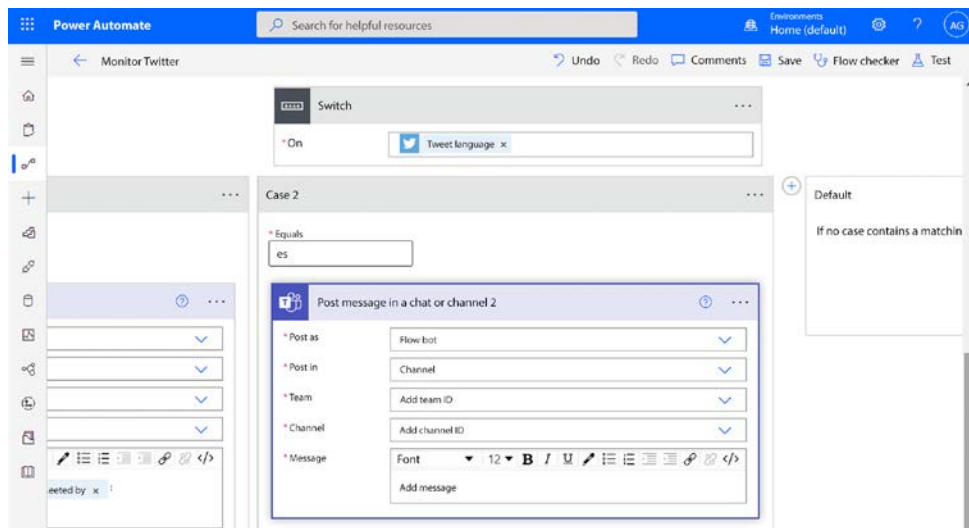


Figure 8.19: Configuring the action



12. Finish configuring the **Team**, **Channel**, and **Message** parameters. In this example, we updated the original text notification to display in Spanish:

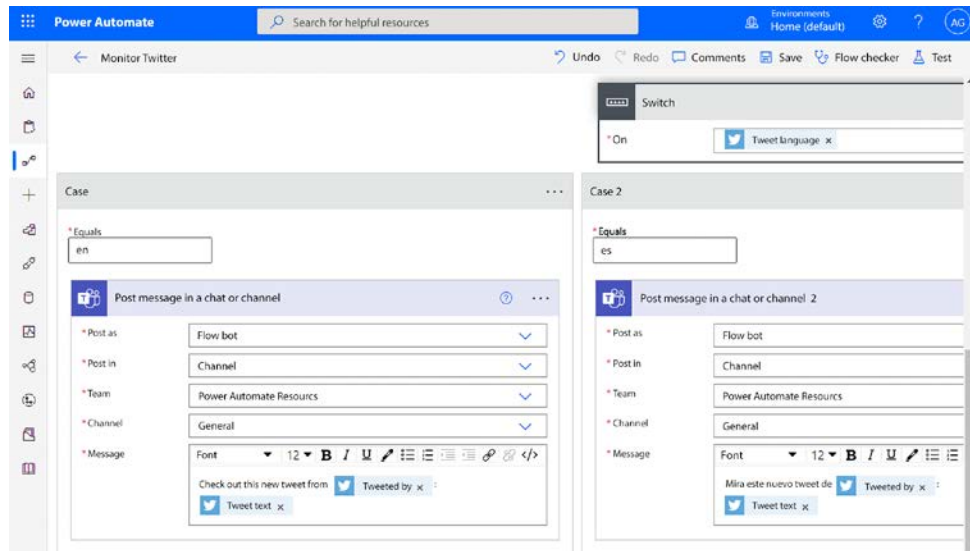


Figure 8.20: Finishing the Case 2 action configuration

13. Scroll to the **Default** case. This is the set of actions that will execute in the event the tweet language can't be determined or doesn't match one of the configured values (it's unlikely in this scenario, since the previous condition groups are already evaluating the tweet language).
14. Click **Add an action**:

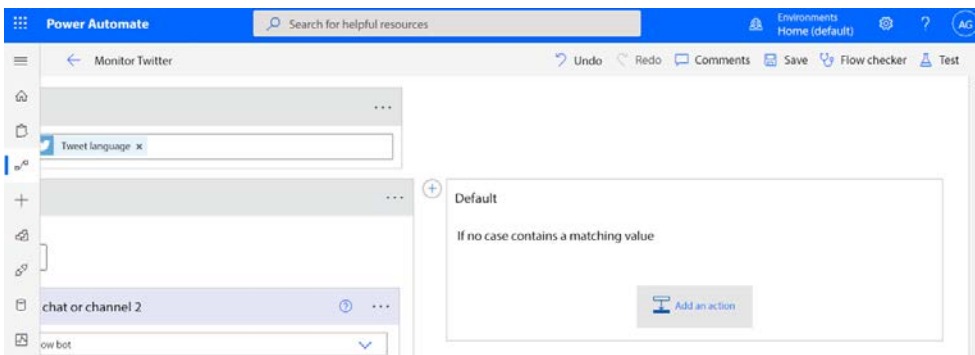


Figure 8.21: Configuring the default action

15. Add a **Post message in a chat or channel** action.

16. Configure the action using the settings of your choice:

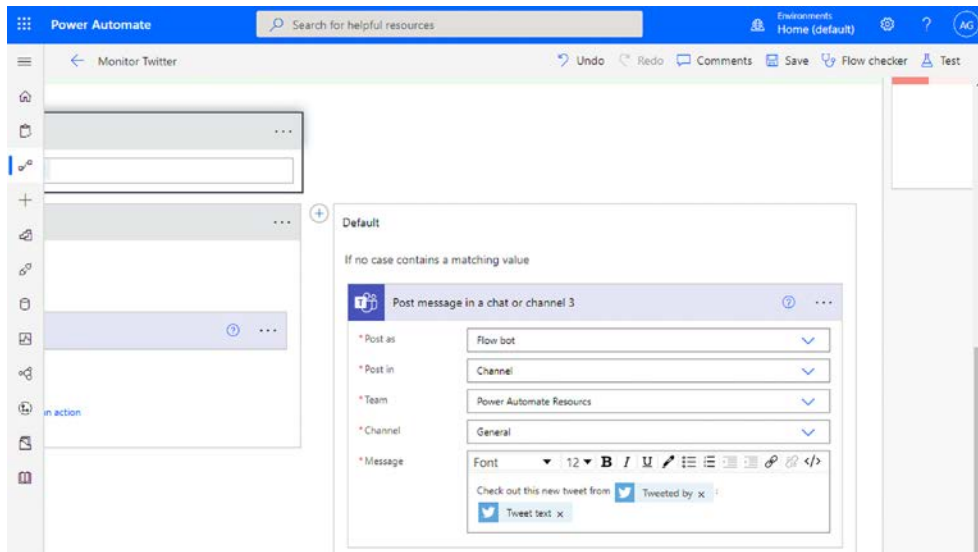


Figure 8.22: Configuring the Default action

17. Click **Save** to complete the flow.

You can test the flow by posting a tweet that meets the logic conditions that you have configured. If you send the tweet in English, the English text should be posted to the channel. If you configure Twitter to use Spanish as your language, the Tweet should show up with the corresponding Spanish text you entered.

How is this different or better than a condition group or using multiple condition statements? In some instances, there may be no objectively correct answer. Flows can be configured to work with a variety of conditional logic methods. Like many development activities, the best practice frequently is to use a method that meets the requirement in the simplest fashion – making it easier for those who come after you to understand your intent and design.

## Summary

In this chapter, you learned how to work with advanced conditions. Advanced conditions allow you to use multiple dynamic content tokens and expressions in order to restrict what data entities are processed by a flow. Using the AND and OR operators, you were able to combine multiple conditions into groups to create a complex decision process for the flow.

You were also introduced to switch conditions – a type of control action that allows your conditional logic to take different paths depending on a broader set of evaluations.

Conditional processing is important because it can be used to help refine and filter data inputs as well as ensure that business criteria or requirements are met prior to flow execution. Conditional processing also gives you an opportunity to create flows that meet complex business criteria.

In the next chapter, we'll begin working with approval workflows. Approval workflows are one of the most popular types of flows for organizations and can even be incorporated into Microsoft Teams.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 9

## Getting Started with Approvals

The flows you've worked with up to this point are largely automated (like detecting new files or emails) or triggered by a user action (such as the button flow to send a template email). In this chapter, we're going to begin exploring flows that require additional user action to continue, such as an *approval* flow.

Approval flows require a special service that's used to hold state information pertinent to the flow. **Dataverse** (formerly the **Common Data Service**) is the foundation for more complex flows such as approvals.

Specifically, this chapter will cover the following topics:


- Understanding Dataverse
- Creating an approval flow
- Responding to approvals

By the end of this chapter, you'll have some introductory knowledge of Dataverse and be able to create basic approval flows.

Let's dive in!

### Understanding Dataverse

**Microsoft Dataverse** (formerly **Common Data Service** or **CDS**) can be thought of in similar terms to a database. It is comprised of data objects called entities. Compared to a database, entities can be described in terms of *database tables*. Whereas database tables have the concept of *columns*, the corresponding object in an entity is called an *attribute*.



Microsoft documentation currently has two separate descriptions for the types of objects stored in Dataverse. We covered the new nomenclature in *Chapter 1, Introduction to Power Automate*. In some cases, however, Dataverse terminology may change, depending on the user interface, protocol, or technologies being used.

One of the biggest terminology changes is the renaming of **entity** to **table**. Since *table* has distinct database implications for this book, we'll use the old terminology of *entity*. Many of the UI components that you'll encounter will still reference Dataverse elements by their previous names.


Dataverse has a number of standard entities that cover a lot of usage scenarios. Some entities, such as activities, pull data from multiple tables as well. There are standard entities in several groups or categories, as listed here:

Entity Type	Example Entities	Description
Customer	Account, Contact, CustomerAddress	Entities that are used for managing and referring to customer account information, such as individuals and addresses.
Activity	Email, Task, Fax, PhoneCall, ActivityParty, ActivityPointer	Entities that are used when users perform interactions or activities with customers, such as leaving phone messages or sending emails.
Annotation	Note	Captures note details (text, attachments) related to a record in Dataverse.
Calendar	Appointment	Calendar entities can be used to keep track of the schedule and availability of a service or resource.
Queue	Public, Private	Queues are used to manage and organize how activities or tasks are processed for a particular customer. Queues are containers for multiple entities, such as tasks.
Subject	Knowledgebase, Incident	Subject entities are used to categorize records, such as sales artifacts, in a hierarchical fashion.
Category	Category	The Category entity can be used to tag records to aid in searchability and discoverability within the system.

<b>Feedback</b>	Feedback	The Feedback entity enables users to store data related to customer feedback for an object, such as a product or service.
<b>Template</b>	DocumentTemplate, KbArticleTemplate, MailMergeTemplate	Template entities help organizations ensure consistency across other entities created in the system.

Table 9.1: Dataverse entity types

Along with these standard entities, Dataverse allows for the creation of custom entities to support specific business processes or goals.



For a full list of entities and the actions that can be performed against them, see the Entity reference: <https://docs.microsoft.com/en-us/powerapps/developer/data-platform/reference/about-entity-reference>.

As it relates to an approval flow, Dataverse is used to store state information in entities throughout the flow. Without this data storage capability, the approval flow would not have a mechanism to track the status or state of any of the approval branches.

While approval flows rely on Dataverse to operate, end users are not responsible for interacting with Dataverse directly to create approval flows. All of the integration, read, and write operations regarding the Dataverse transactions are handled internally, without the user needing to work with them.

Next, we'll work through the steps used to create an approval flow.

## Creating an approval flow

Approval flows can cover a number of scenarios, such as requesting time away, manager sign-off on an expense report, or business owner sign-off on product marketing materials that are to be published. Approval flows typically fall into two categories:

- **Automated**, such as when a new item is added to a SharePoint document library or list. You might use an automated flow to start the approval process for an expense report, which is triggered when a document is uploaded to a site.
- **Instant**, such as when a user manually starts an approval workflow from an existing document, list item, or application. You might configure an instant approval to start a process for approving an individual piece of content for external publication.

Regardless of the use case and configuration, an approval flow will require the Approvals connector. The Approvals connector has a number of actions and object types that the flow can interact with, though mostly, we'll be working with *responses* and *outcomes*.



You can learn more about the Approvals connector here: <https://docs.microsoft.com/en-us/connectors/approvals/>.

When starting an approval, the workflow process sends an email to the approver and adds an item to the approver's queue. For the requester, the **Approvals** node shows actions that have been initiated or sent. Approvals may be responded to via email, the Teams Approvals app, or the Approvals section in the Power Automate portal.

In this example, we're going to create a sample approval for a vacation request. This flow has the following prerequisites:

- Azure Active Directory user identities with the **Manager** property need to be populated.
- A SharePoint Modern Team Site with a modern list, including two date/time columns for **Start** and **End** dates.

In the upcoming sections, we'll set up an environment that supports the approval and then create the approval flow.

First, we'll create the SharePoint site and list that will be used in the flow.

## Creating a SharePoint site and list

In this instance, the requests will be added to a SharePoint list. You can create a site with a list by following these steps:

1. Navigate to the Microsoft 365 Admin Center (<https://admin.microsoft.com>), log in with an administrative identity, and navigate to the SharePoint Admin Center (**Admin centers | SharePoint**).
2. Select **Sites | Active sites** and then click **Create**.
3. Select **Team site**:

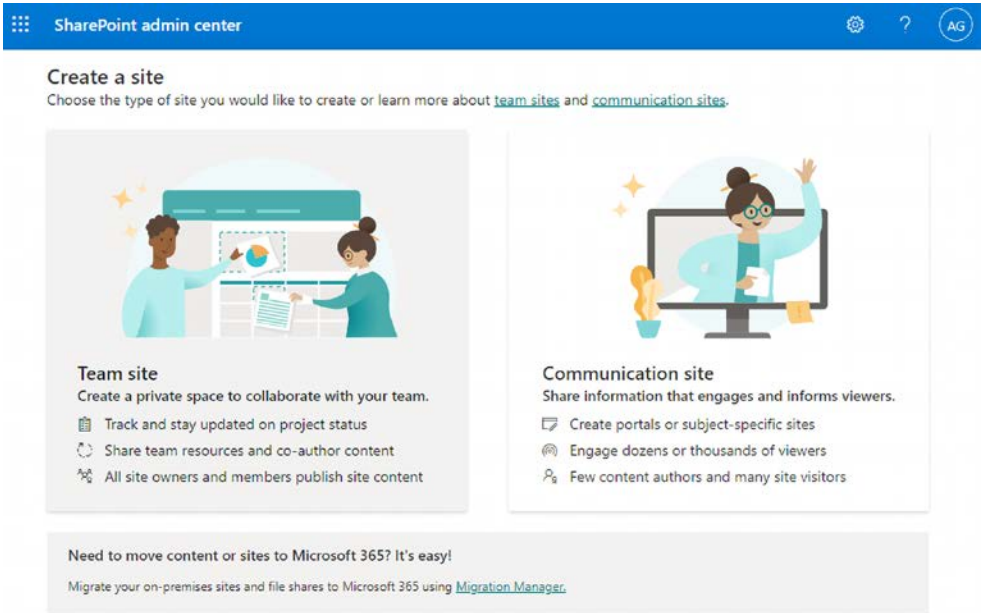


Figure 9.1: Creating a new Team site

4. Fill in the properties for **Site**, **Owners**, **Language**, **Privacy**, and **Time zone**. Then click **Next**.
5. Add any additional members or owners and click **Finish**.
6. Navigate to the new site.
7. From the site's navigation menu, select **Site contents**.
8. Click **New | List**:

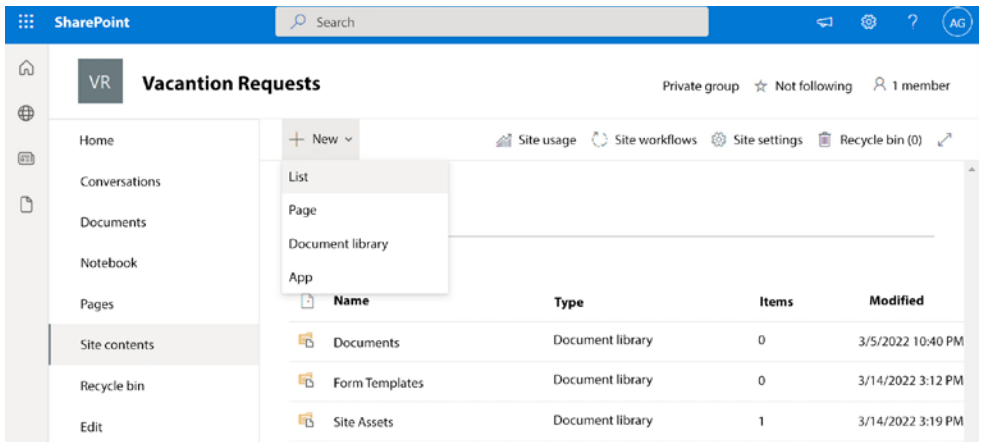


Figure 9.2: Creating a new list for vacation requests



- 9. Select **Blank list** as the template type. Enter a name for the list and click **Create**.
- 10. Click **Add column**, select **Date and time**, and then create a column for the start date (such as **From**):

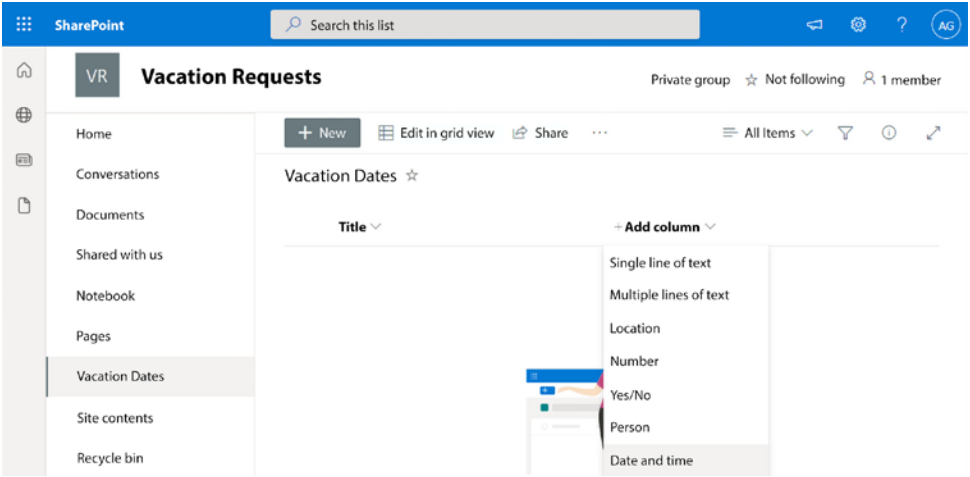


Figure 9.3: Adding columns

- 11. Click **Add column**, select **Choice** as the type, and then set the name as Status. Click the **choice** options and rename them to Pending, Approved, and Denied, as shown in Figure 9.4. Set the default value to **Pending** and click **Save** when finished:

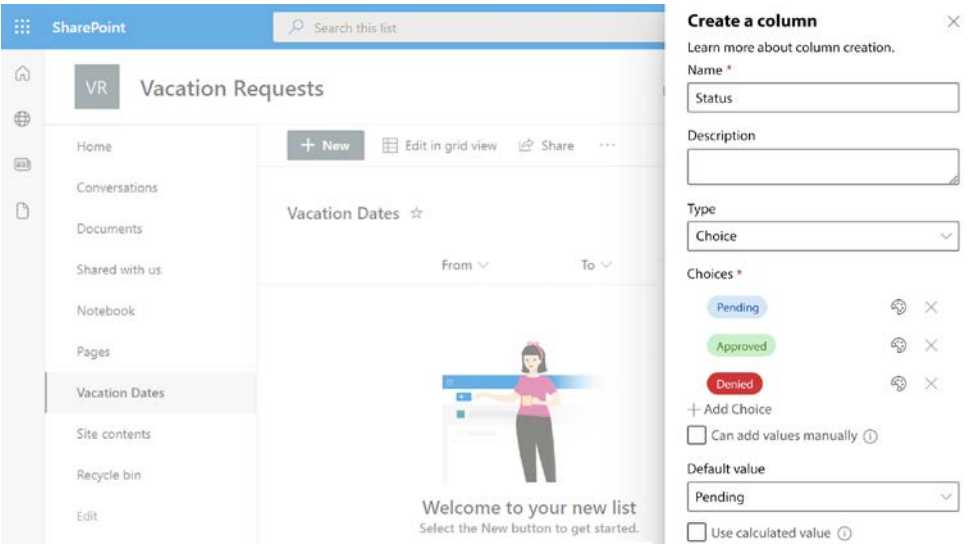


Figure 9.4: Configuring the Choice column

12. Add additional columns based on the following table (and any additional columns for details you'd like):

Name	Type	Possible Values	Default Value
To	Date and time	[NA]	[NA]
Details	Multiple lines of text	[NA]	[NA]

Table 9.2: List column settings

Your SharePoint site and list are now ready to take vacation requests. Next, we'll work on creating the actual approval flow.

## Creating an approval

Once the SharePoint site and list have been created, we'll need to create the actual approval that will process requests as they appear on the list. Follow these steps to create the approval flow:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **Create** and, under **Start from blank**, select **Automated cloud flow**.
2. Enter a flow name, such as Vacation Approval. Search for the **When an item is created** SharePoint trigger. Select it and click **Create**.
3. Inside the **When an item is created** trigger, select the SharePoint site you created in the prerequisite step for the **Site Address** value and select the SharePoint list you created in the prerequisite step for the **List Name** value. If the list doesn't show up, use the **Custom value** option in the dropdown to manually specify the address:

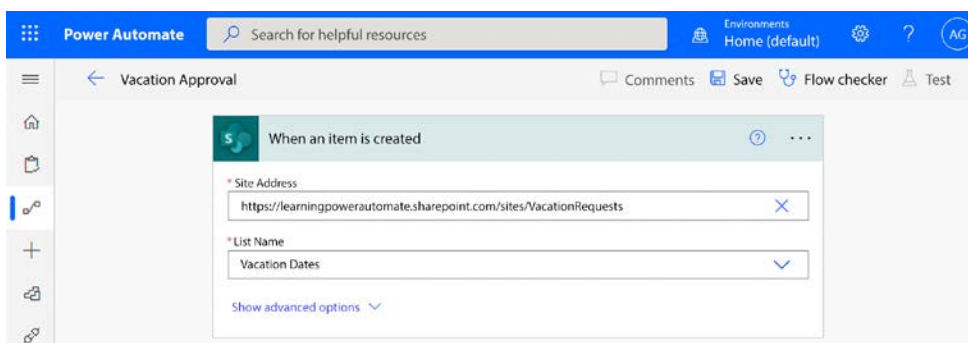


Figure 9.5: Selecting the SharePoint site and list to monitor

4. Click **New step**. Add the **Get my profile (V2)** action.
5. Click **New step**. Add the **Get manager (V2)** action.

6. In the **User (UPN)** value box of the **Get manager (V2)** action, add the dynamic content token for **User Principal Name** under the **Get my profile (V2)** section:

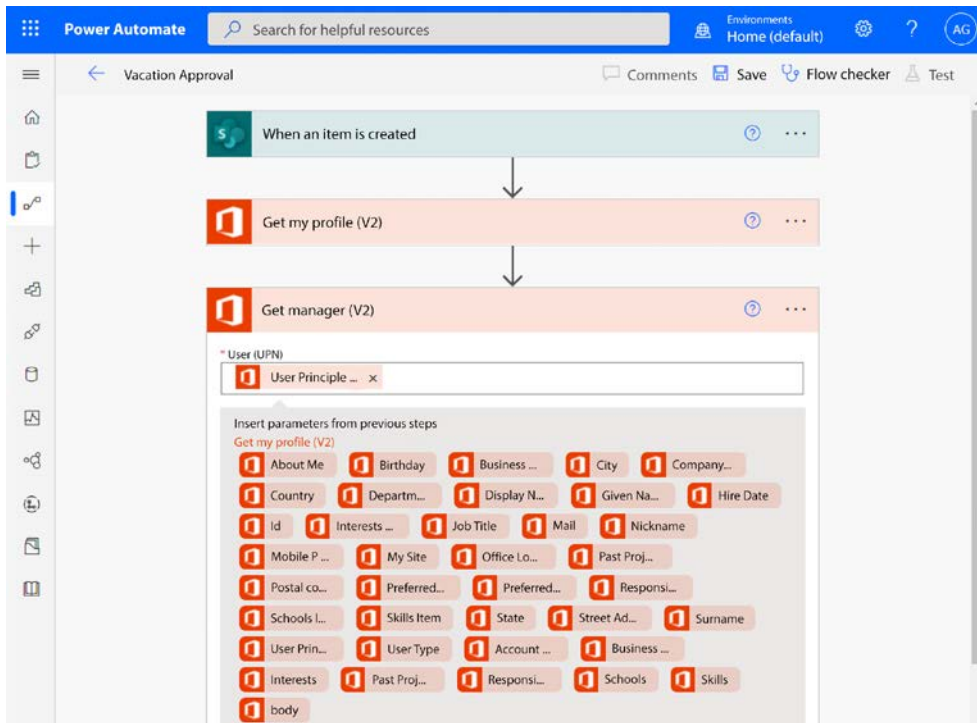


Figure 9.6: Adding the User Principal Name dynamic content value to the Get manager (V2) action

7. Click **New step** and add the **Start and wait for an approval** action:

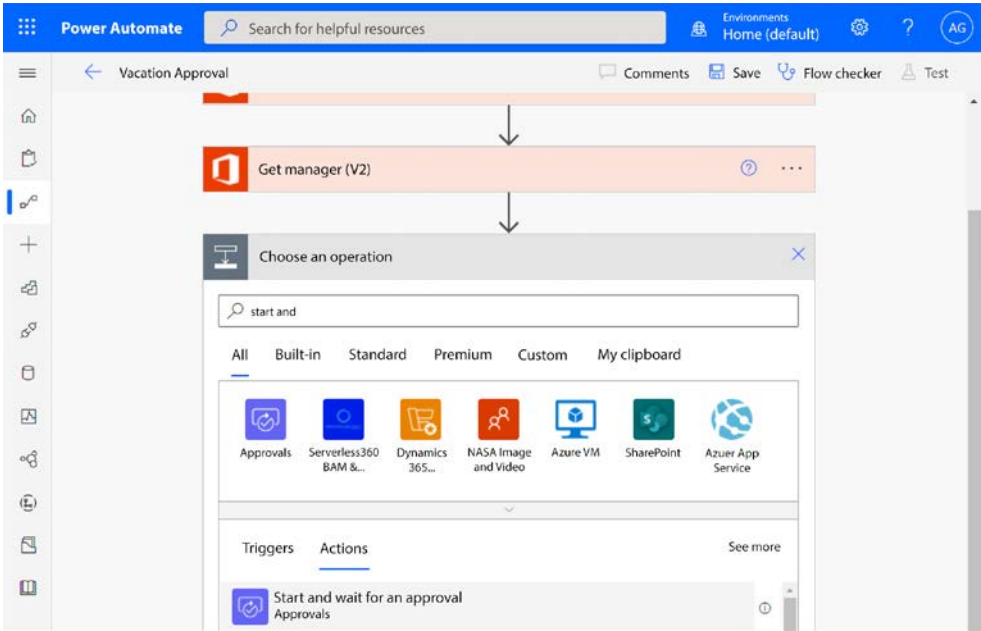


Figure 9.7: Adding an approval action

8. In the **Approval type** box, select the **Custom Responses - Wait for one response** option:

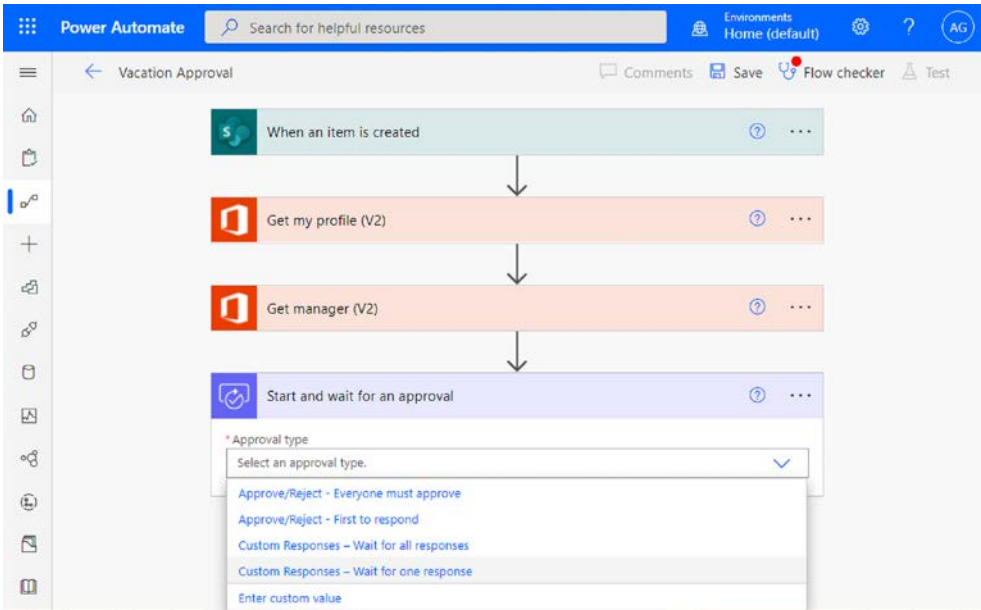


Figure 9.8: Selecting the Custom Responses approval type

9. Fill out the form as appropriate. For **Response options item**, enter the text Approve. Then, select **Add new item** and add another option for Deny. These are the options that will be displayed when the approval is sent to the approver:

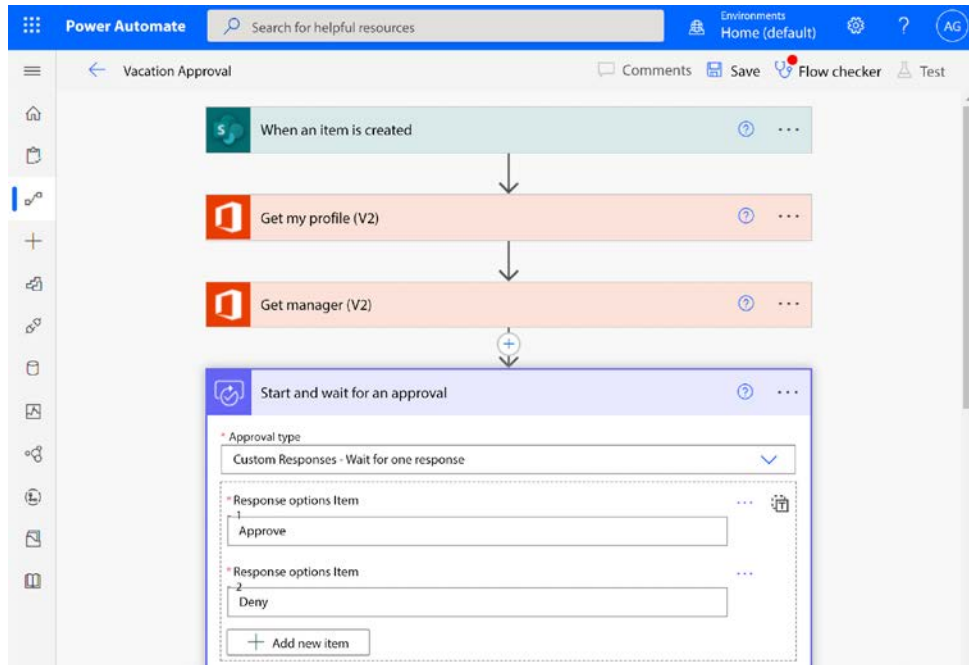


Figure 9.9: Configuring response options

10. In the **Title** box, you can create a title that will be displayed for the approval notification. You can also use dynamic content tokens to populate it (such as the **Display Name** dynamic value under the **Get my profile (V2)** action).

11. In the **Assigned to** box, select the **Mail** dynamic token under the **Get manager (V2)** section:

The screenshot shows the 'Vacation Approval' flow configuration in Power Automate. The 'Custom Responses - Wait for one response' section has two response options: 'Approve' and 'Deny'. Below this, the 'Title' field is set to 'Vacation Request for' followed by a dynamic token 'Display Name'. The 'Assigned to' field is set to a dynamic token 'Mail'. A section titled 'Insert parameters from previous steps' shows a list of dynamic tokens from the 'Get manager (V2)' action, including 'About Me', 'Birthday', 'Business ...', 'City', 'Company...', 'Country', 'Departm...', 'Display N...', 'Given Na...', 'Hire Date', 'Id', 'Interests...', 'Job Title', 'Mail', and 'Nickname'.

Figure 9.10: Configuring the Title and Assigned to fields of the approval action



If you are developing and testing your flow in a production environment, you may wish to add your own email address as the target as opposed to the mail dynamic content token from the **Get Manager** action (so you don't generate a lot of test messages for your manager).

12. In the **Details** box, enter any additional details or text you wish to include in the approval notification.

13. In the **Item link** box, add the **ID** dynamic content token under the **When an item is created** section:

The screenshot shows the 'Start and wait for an approval' action in Power Automate. The configuration includes:

- Approval type:** Custom Responses - Wait for one response
- Response options Item 1:** Approve
- Response options Item 2:** Deny
- Title:** Vacation Request for **Display Name**
- Assigned to:** Mail
- Details:** Markdown supported (see <https://aka.ms/approvaldetails>)
- Item link:** ID
- Item link description:** Describe the link to the item

A blue arrow points to the 'Item link' field, which contains the 'ID' dynamic content token.

Figure 9.11: Configuring the approval action

14. Click **New step** and add the **Condition** control.
15. In the **Choose a value** box, add the **Outcome** dynamic content token from the **Start and wait for an approval** action:



If you select one of the **Responses** tokens, you can still continue, but your action may be wrapped in an **Apply to each** container, which will require you to do a little extra legwork.

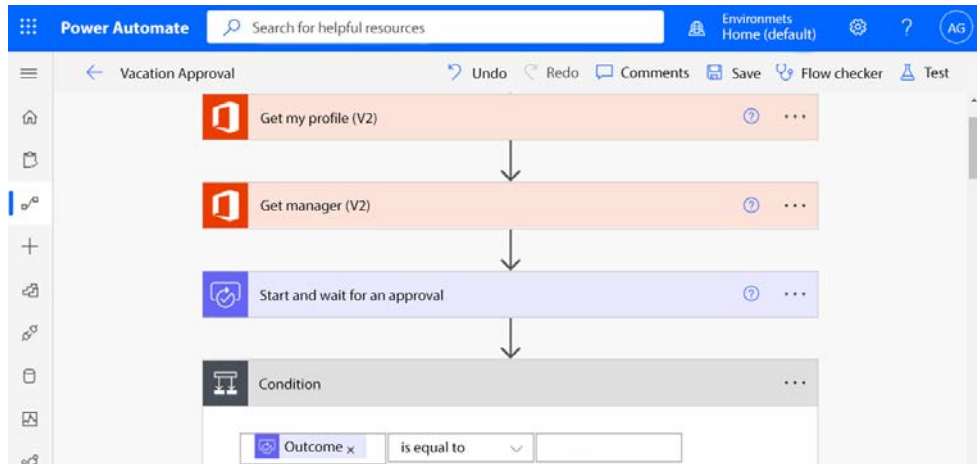


Figure 9.12: Adding the Responses dynamic content value

16. In the operator box, ensure **is equal to** is selected. On the right-hand side, enter the Approve value. This text should match the **Response options** text for the **Approve** option we entered in *step 9*:

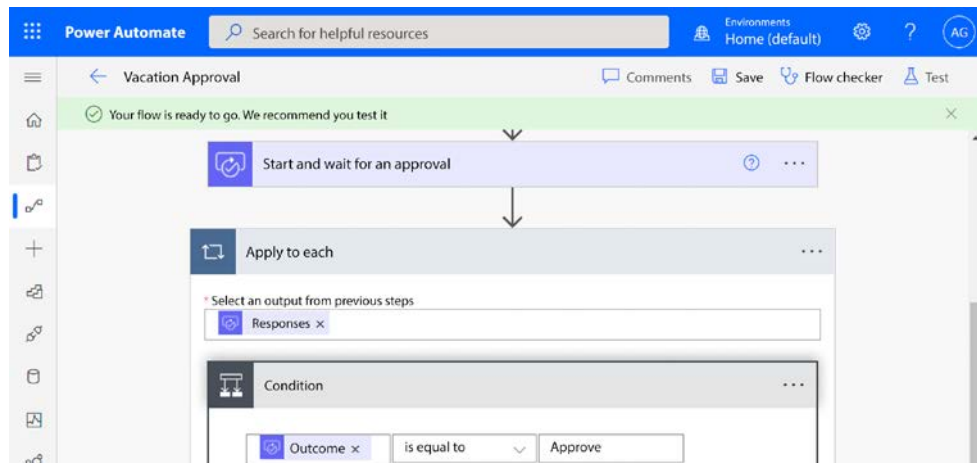


Figure 9.13: Configuring the Condition control

17. Under the **If yes** branch of the condition, click **Add an action**.
18. Select the SharePoint **Update item** action.
19. On the card for the **Update item** action, in the **Site Address** box, choose the **Vacation Approvals** site. If the site is not populated, you may need to select the **Enter custom value** option and enter the site manually.



20. On the card for the **Update item** action, in the **List Name** box, select the list containing the vacation request item.
21. On the card for the **Update item** action, in the **Id** box, select the dynamic content token for **ID** under the **SharePoint When an item is created** section.
22. On the card for the **Update item** action, in the **Title** box, select the dynamic content token for **Title** under the **When an item is created** section.
23. On the card for the **Update item** action, in the **Status Value** box, select the **Approved** value:

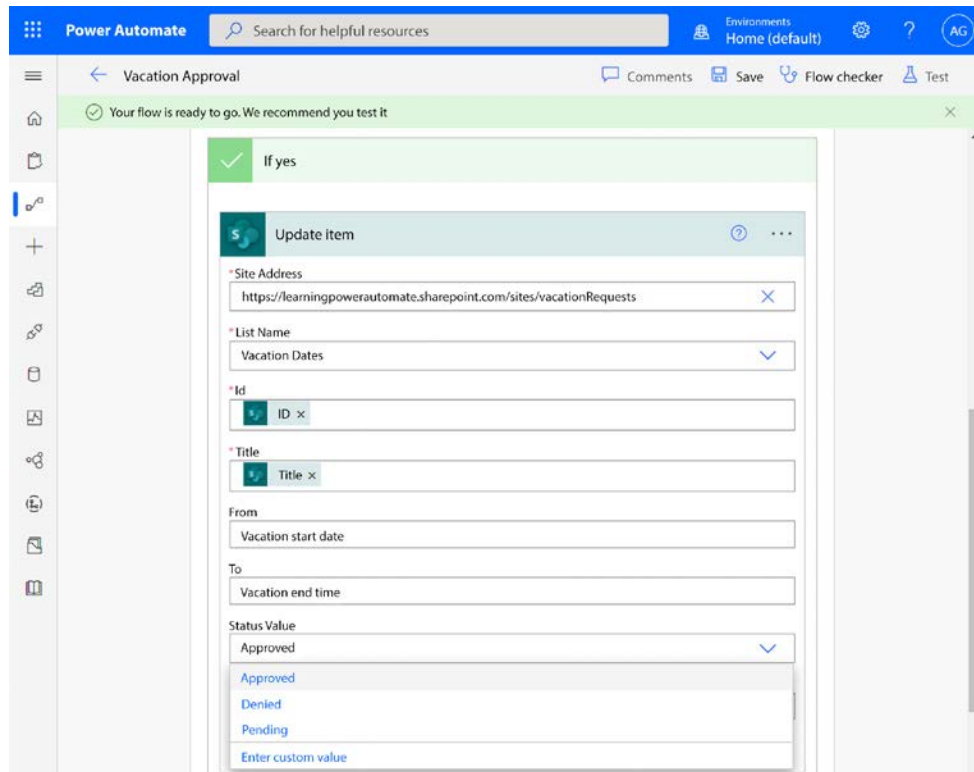


Figure 9.14: Configuring the Update item action in the If yes branch

24. Under the **If no** branch, repeat *steps 17* through *22*.
25. On the card for the **Update item** action in the **If no** branch, in the **Status Value** box, select the **Denied** value.

26. Review your flow to ensure it looks similar to the example in *Figure 9.15*:

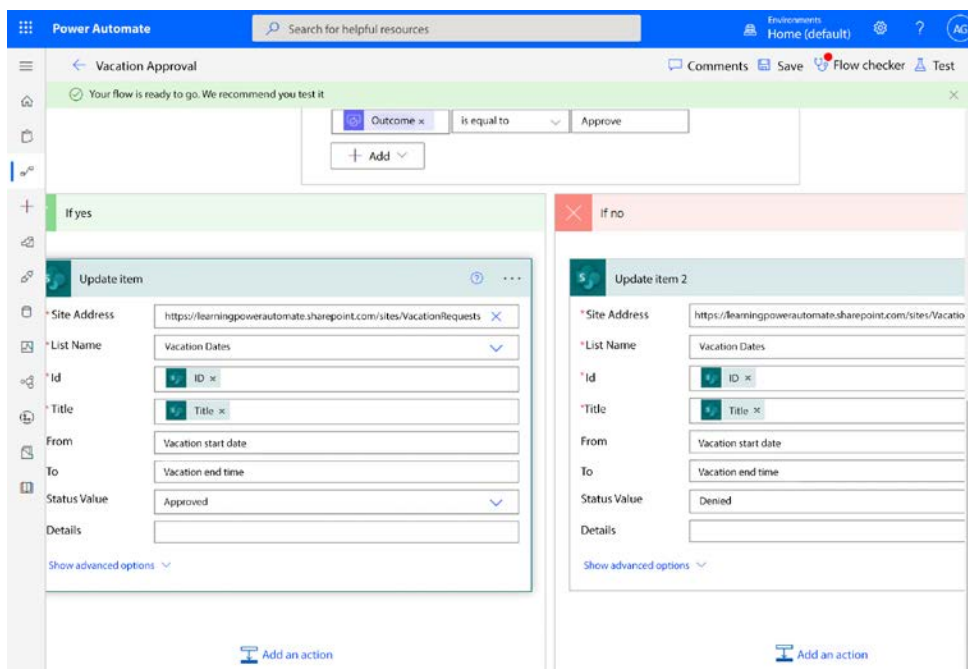


Figure 9.15: Reviewing the configuration

27. Click **Save** to save the flow.

For an advanced modification to this, you can also add individual or group calendar events, as you'll see in the next section.

## Adding calendar events and notifications

You can increase this approval's usefulness by adding automatic calendar appointments to the **If yes** branch. You can also include additional information in emails to the request originator if desired. Use the following optional steps to modify and extend this approval with a shared calendar appointment:

1. With the **Vacation Approval** flow open, navigate to the end of the **If yes** branch.

2. Click **Add an action** and select the **Create a group event (V2)** action from **Office 365 Groups**:

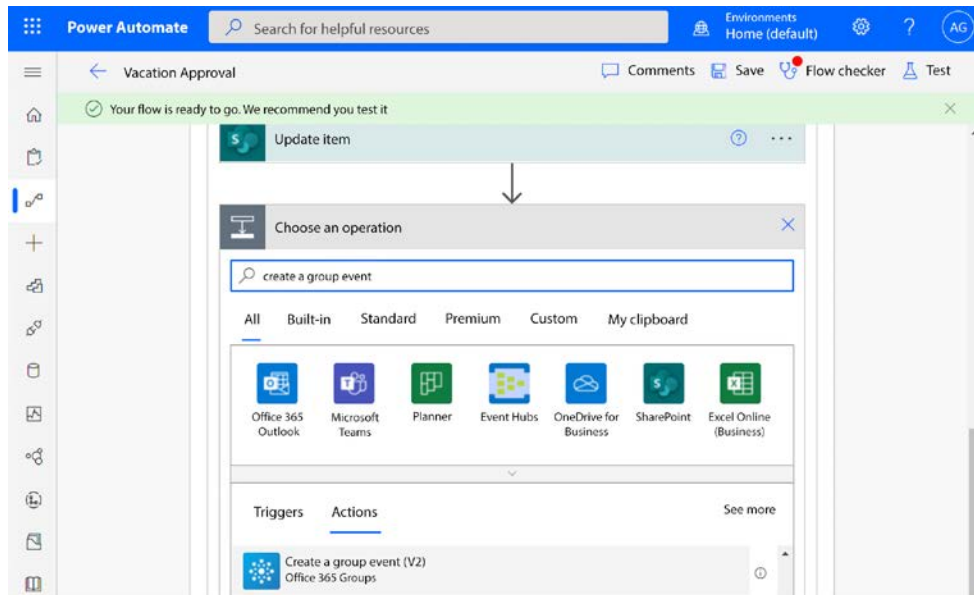


Figure 9.16: Adding a group event action

3. On the **Create a group event (V2)** card, in the **Group Id** box, select the Microsoft 365 group corresponding to the SharePoint site for the vacation's approval. You can also select another group as long as the connected account has access.
4. On the **Create a group event (V2)** card, in the **Subject** box, enter any value consisting of a combination of dynamic content tokens and static content, such as the **Create by Display Name** dynamic content token from the SharePoint **When an item is created** section.
5. On the **Create a group event (V2)** card, in the **Start Time** box, select the dynamic content token representing the start time from the SharePoint **When an item is created** section. If you don't see the dynamic content values displayed, you may need to adjust your browser zoom level.
6. On the **Create a group event (V2)** card, in the **End Time** box, select the dynamic content token representing the end time from the SharePoint **When an item is created** section. If you don't see the dynamic content values displayed, you may need to adjust your browser zoom level.

- On the **Create a group event (V2)** card, expand the advanced options. Set the **Is All Day** field to **Yes** and the **Show As** option to **Free**. Edit any additional options as per your preferences:

The screenshot shows the 'Create a group event (V2)' card in the Power Automate interface. The card is titled 'Vacation Approval' and has a green status bar indicating 'Your flow is ready to go. We recommend you test it'. The card contains the following fields:

- Subject:** Created By Dis... - Time Away
- Start Time:** From x
- End Time:** To x
- Body:** Font 12, Bold, Italic, Underline, Link, Unlink, Text color, Background color, Code
- Location:** Location of the event.
- Importance:** The importance of the event: Low, Normal, or High.
- Is All Day:** Yes
- Is Reminder On:** Set to true if the event has a reminder.
- Reminder Start Duration:** Time in minutes before event start to remind.
- Show As:** Free
- Response Requested:** No

A 'Hide advanced options' link is located at the bottom of the card.

Figure 9.17: Configuring the update item action

- Click **Save** to save the flow.

The Microsoft 365 group calendar will now be updated with a calendar item showing the requester's approved time off.



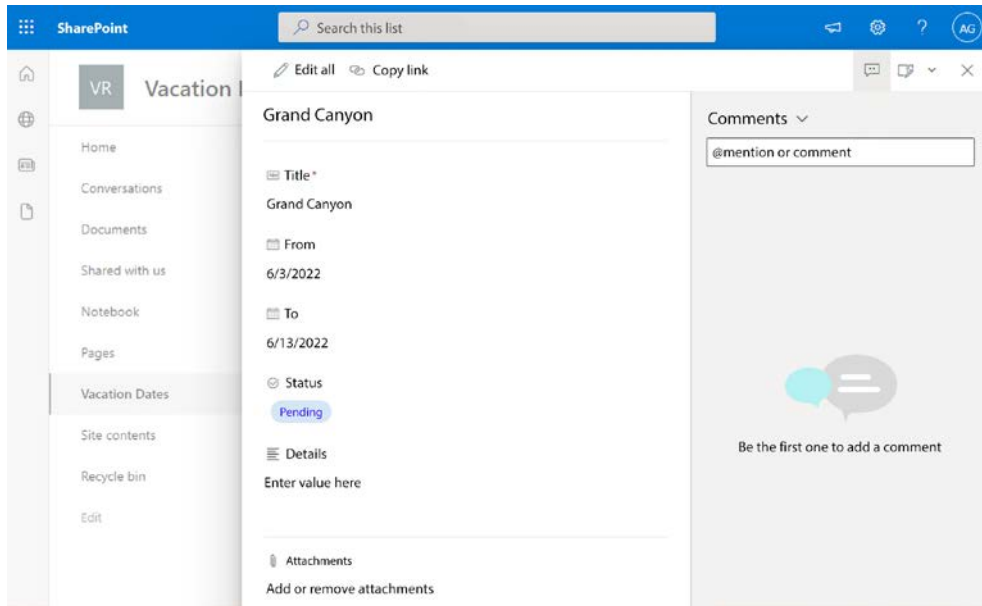
Extra credit: In addition to creating an entry on a Microsoft 365 group calendar upon approval, you can also add another action to configure a calendar event on the requester's calendar.

Next, we'll learn how the flow works as a requester.

## Starting the flow

The easiest way to test this flow is to start a vacation request. To do so, follow these steps:

1. Navigate to the SharePoint list that was created as part of this exercise.
2. On the list, click **New** to create a new request item.
3. Fill out the form. Add a title, select the start and end dates, and then click **Save**:



The screenshot shows a SharePoint interface for a list titled 'Vacation Requests' (VR). The left sidebar contains navigation links: Home, Conversations, Documents, Shared with us, Notebook, Pages, Vacation Dates (selected), Site contents, Recycle bin, and Edit. The main content area displays a form for a new item titled 'Grand Canyon'. The form fields include: 'Title\*' with the value 'Grand Canyon'; 'From' with the date '6/3/2022'; 'To' with the date '6/13/2022'; and 'Status' with a dropdown menu showing 'Pending'. Below these fields is a 'Details' section with a placeholder 'Enter value here'. At the bottom, there is an 'Attachments' section with the text 'Add or remove attachments'. On the right side of the form, there is a 'Comments' panel with a search bar '@mention or comment' and a message 'Be the first one to add a comment' accompanied by a speech bubble icon.

Figure 9.18: Creating a new vacation request

4. Click **Save**.

At this point, you have created a SharePoint list that can be used as the trigger for the approval flow, as well as the approval flow itself. Finally, you triggered the flow. Next, we'll see how it looks from the approver's side.

## Responding to approvals

As an approver, you'll receive an email notification stating that action is required. Approvals will also show up in the **Approvals** section of both the Power Automate web portal and the mobile app as well as within the Approvals app in Microsoft Teams. An approval can be processed from any of those locations.

The approval has buttons based on the various **Response option** items selected in the approval flow. In the following screenshot, you can see that the options we configured when creating the approval flow are displayed in the body of the email. The approver can select **Approve** or **Deny**, enter any optional information, and then click **Submit**.

Figure 9.19 shows what the approval email looks like when it's delivered to the approver:

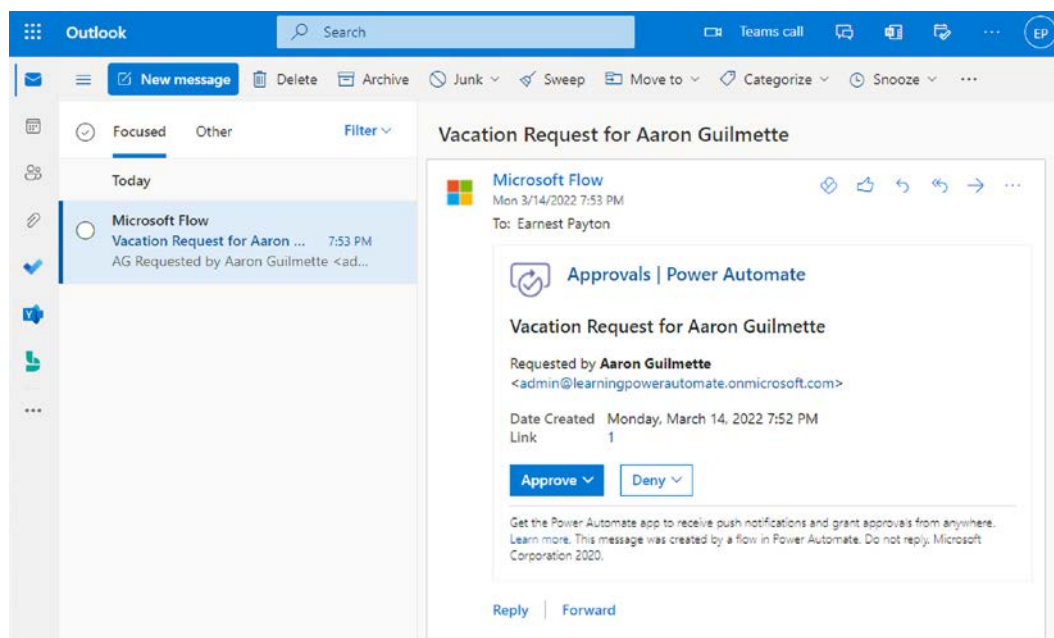


Figure 9.19: Approval request email

If your organization has deployed the Approvals app, the request is also available there, as shown in *Figure 9.20*:

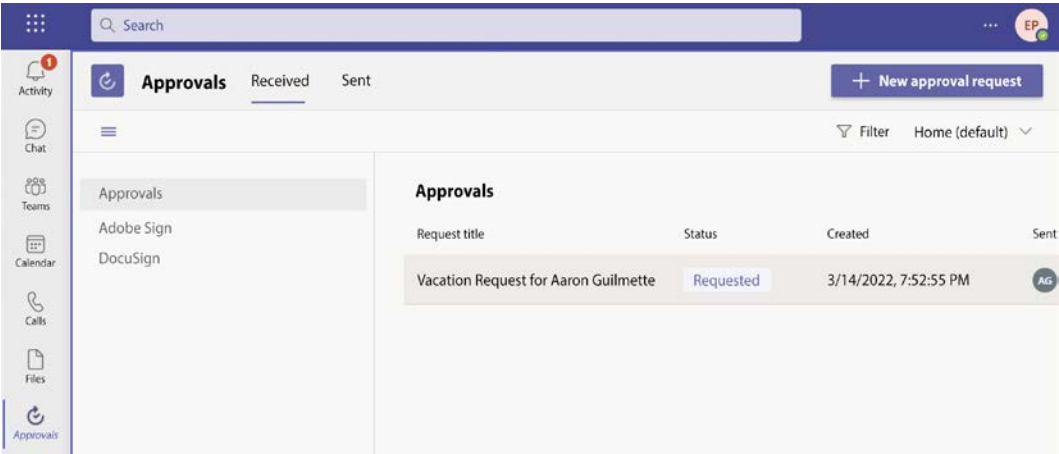


Figure 9.20: Approvals app in Microsoft Teams

Finally, the approver can also log in to the Power Automate web portal and navigate to **Action items | Approvals** to see and act on any pending approval requests:

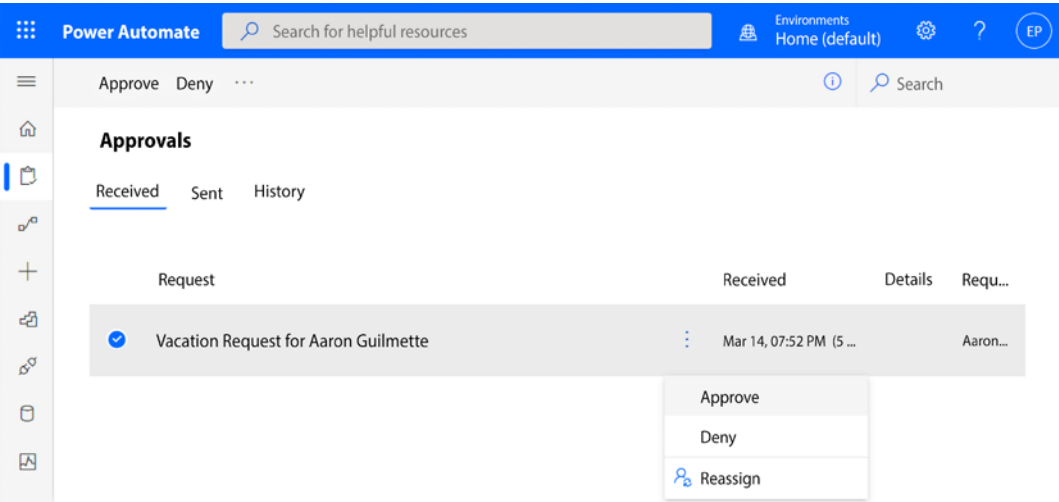


Figure 9.21: Reviewing approvals in the Power Automate web portal

As you can see, approvals can be accessed and processed from any environment. Because approvals are surfaced in multiple locations, an approver may not even have to leave their application context and can continue working with minimal interruption.

After choosing an option, the item is updated in the SharePoint Online list, as shown in the following screenshot:

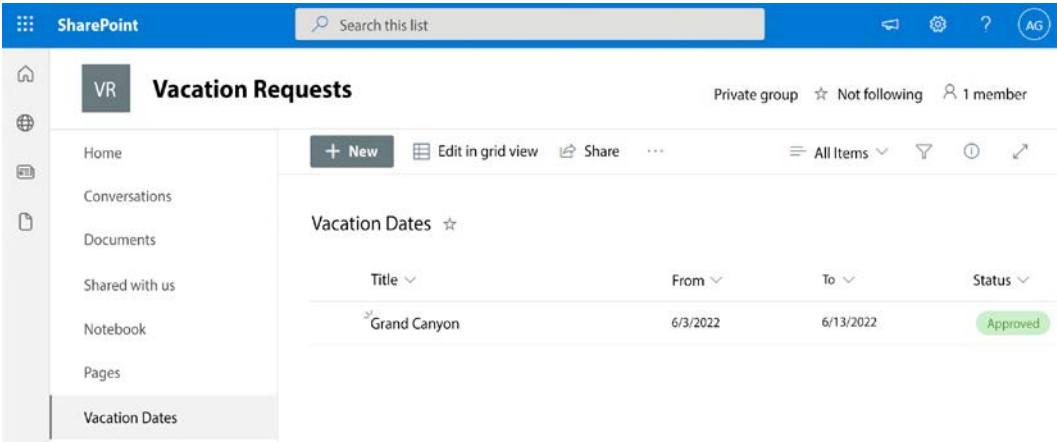


Figure 9.22: Viewing the updated SharePoint list item

If you updated the flow with the group calendar event action, you will also see the calendar item through the Outlook web app:

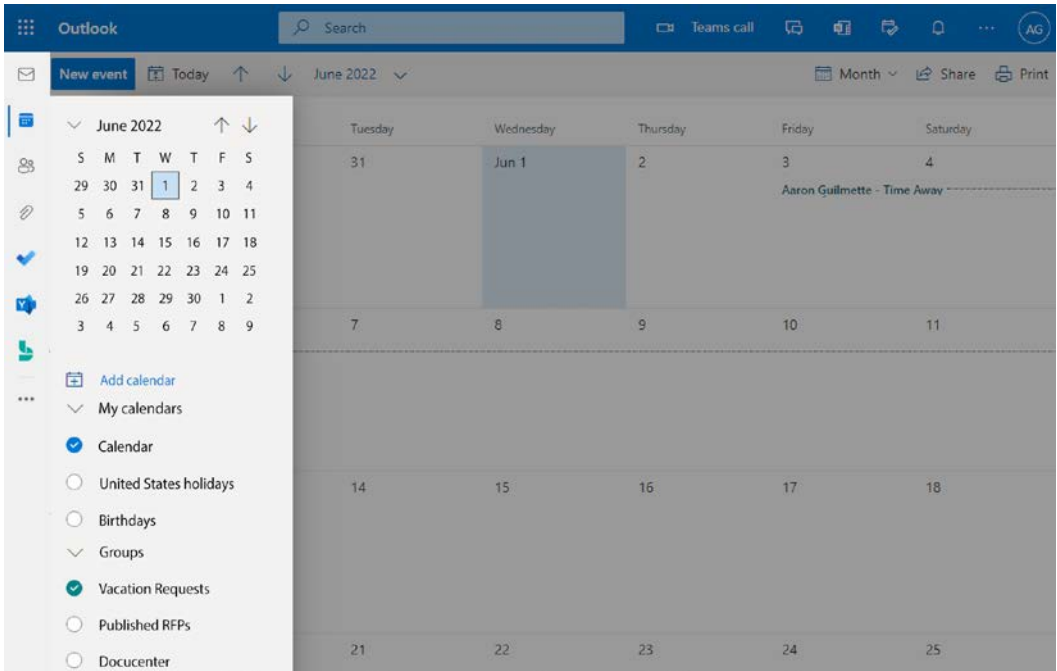


Figure 9.23: Viewing the Vacation Requests group calendar



You can continue editing and refining the approval request flow by adding email notifications, attachments, or even integrating it into other flows.

## Summary

In this chapter, you learned about one of the most popular and useful flow capabilities available in Power Automate – the approval flow. Approval flows frequently utilize Dataverse (formerly called CDS) to store state information. Approval flows can be used for simple tasks such as vacation requests or expense forms, and can also be integrated into more complex tasks such as document publishing and content automation. Finally, you learned how to create an approval flow, trigger the approval, and respond to it as an approver.

In the next chapter, we'll begin using Power Automate to interact with Microsoft Forms.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 10

## Working with Multiple Approvals

In *Chapter 9, Getting Started with Approvals*, you learned the basic mechanics of an approval flow with a single approver. While that scenario is very common, it doesn't address all of the likely requirements that you'll encounter. In many instances, you may need to have multiple individuals approve items, either in sequence or in parallel. This is where multiple approvals fit in.

In this chapter, we'll discuss some common approval scenarios and then create an example flow using this knowledge:

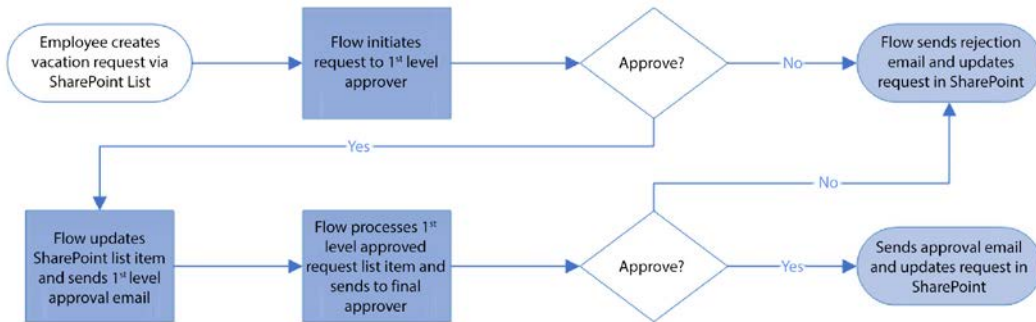
- Working with sequential approvals
- Working with parallel approvals
- Working with advanced scenarios
- Creating a basic sequential approval
- Adding parallel branches

By the end of this chapter, you'll understand how you can use these techniques independently or together to make more complex approval workflows.

### Working with sequential approvals

A sequential approval is an approval scenario requiring various stages of approval before a final approver can accept or reject it. Common scenarios include expense reports or vacation requests where an immediate manager approves an item, which is then routed to a second-level manager.

An example sequential approval sequence is shown in *Figure 10.1*:



*Figure 10.1: Sequential approval*

The figure depicts this order of events:

1. The user adds an item to a SharePoint list.
2. Power Automate, which is configured to use a new item appearing on the list as a trigger, initiates the approval flow and sends the approval request to the first approver.
3. The first approver responds to the request (either *Approves* or *Rejects*).
4. The Power Automate flow processes the approval answer. However, unlike a single approval ending the flow, Power Automate instead starts the second stage (or final stage) of the approval.
5. The second approver responds to the request (either *Approves* or *Rejects*).

An approval process can have any number of approval actions and is not just limited to the first and second stages.

Next, we'll look at an overview of parallel approvals.

## Working with parallel approvals

Whereas sequential approvals process as a single thread in a row or stages, parallel approvals happen independently. A vacation request may also take the form of a parallel approval: you may need two individuals to approve time off (such as your immediate manager as well as an HR representative or a direct manager and a project manager), but those individuals might be in different management or reporting structures and their approval processes are independent of each other.

Parallel approval processes allow different branches of approvals to happen independently. Logic at the end of the flow is used to evaluate whether the overall approval is successful.

## Working with advanced scenarios

There are some additional configuration options that are available for creating even more complex approval workflows. These options are sometimes used in large organizations where one or more individuals or groups can interact with the approval process. Two such options we'll briefly look at include **mixed approval types** and **everyone must approve** options.

### Mixed approval types

For very complex scenarios, an approval may contain multiple levels of sequential approvals, a mix of parallel and sequential approvals, or even a sequential approval inside a parallel approval branch. These types of approvals are known as **mixed approvals**. When evaluating outcomes and responses, you'll need to make sure you're selecting the right dynamic content tokens.

### Everyone must approve

When creating an approval workflow, it may be desirable to only send a final response if one of two conditions is met:

- *Everyone* included in the approval chain approves
- *Any single approver* rejects the approval request

These types of approvals are known as **everyone must approve**.

To demonstrate some of these techniques, we'll create an approval that uses a series (or sequence) of approvals.

## Creating a basic sequential approval

In this section, we're going to create a basic sequential approval that involves first- and second-stage approval. These types of approvals are common in scenarios where an individual requests more than a certain number of vacation days or needs to make an expenditure that reaches a certain threshold. For this example, we'll be creating an approval that evaluates a requested purchase order price. If it's greater than the \$1,000 approval limit of the user's immediate manager, it will trigger another approval sequence.

Like some other workflows in this book, we'll use a SharePoint list as the starting point. You can also take input from other methods, such as reading content from a file, a REST API-based webhook from another system, or even requesting user input (which you'll learn about in *Chapter 14, Accepting User Input*).

This flow will need a list with a number of different columns and data types. The columns will be used to store information as it moves through the process. The overall process will look like this:

- 1. Configuring the prerequisites
- 2. Creating the flow
- 3. Testing the flow

The design of the flow will stipulate that the first-level approver is the requester’s manager, while the second-level approver is a user called Finance Manager. The request will need to be approved by both users in order for it to be marked as *Approved*.

We’ll configure the prerequisites next.

### Configuring the prerequisites

First, you’ll need to configure a SharePoint list, which will be used to hold data for the purchase order request.

Follow these steps to configure the list:

- 1. Navigate to a SharePoint site that will contain the list used to start this workflow. If you don’t have a suitable site, you can create one.
- 2. Select **Site contents** from the navigation menu, and then click **New** and select **List**.
- 3. Create a SharePoint list. Configure it with the following columns and types:

Column Name	Column Type
Title	A single line of text (column exists by default)
Modified	Date and time (column exists by default)
Created	Date and time (column exists by default)
Purchase Amount	Currency
Comments	A single line of text
Pre-approval required	Yes/No (set No as the default)
Pre-approved	Yes/No (set No as the default)
Approved	Yes/No (set No as the default)
Created by	Person or group (column exists by default)
Modified by	Person or group (column exists by default)

Table 10.1: Configuring list columns

Once the columns have been created, you can move on to creating the approval flow. Be sure to note the URL of the SharePoint site and list that you're using, as you'll need them in the next section.

## Creating the flow

Once your SharePoint list has been created, it's time to start working with the flow itself. We'll create this flow in several sections:

- Creating the trigger
- Creating the first-stage approval
- Creating the second-stage approval
- Completing the flow

Let's go!

### Creating the trigger

The trigger will be used to initiate the workflow. This flow's trigger action is a user placing a purchase request in the SharePoint list.

To create the flow, follow these steps:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>), log in, and click **Create**.
2. Select **Automated cloud flow**.
3. Enter a name for the flow (such as **Purchase Order Request Two-Stage Approval**).
4. From the list of triggers, select the **When an item is created** SharePoint trigger, and then click **Create**.
5. Expand the trigger by clicking on the title bar. In the **Site Address** field, select the URL for the SharePoint site where the list is stored. If the site does not appear in the dropdown, select **Enter custom value** and then enter the site address.
6. In the **List Name** field, select the name of the list you created in the *Configuring the Prerequisites* section.
7. Click **New step**.
8. Select the **Get manager (V2)** Office 365 Users action.

9. In the **User (UPN)** field of the **Get manager (V2)** action, add the dynamic content token for **Created by Email** under the **When an item is created** section:

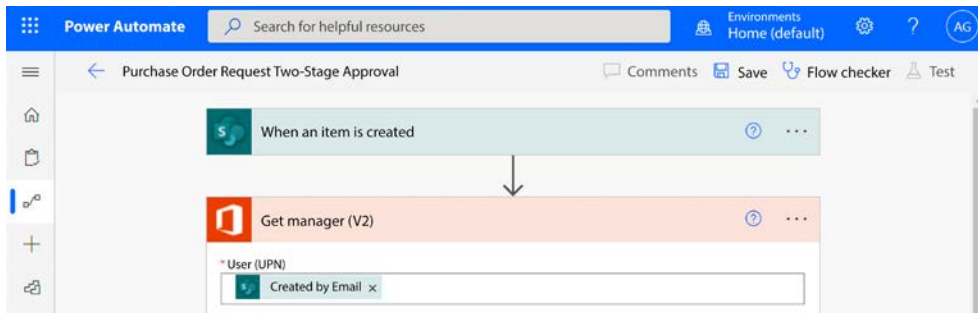


Figure 10.2: Configuring the Get manager (V2) action

10. Click **New step**.
11. Select **Condition** control.
12. On the left side of the condition, select the **Purchase Amount** dynamic content token from the **When an item is created** SharePoint section. Choose **is greater than** for the evaluation operation type. Enter **1000** on the right side of the condition.
13. Click **Save** to save your progress.

Now that the trigger and condition have been created, it's time to proceed to create the first stage of the approval process.

## Creating the first-stage approval

This stage of the approval is routed to the user's immediate manager and can take two paths: if the dollar value is over \$1,000, it will proceed down a branch that will require a second-stage approval. If it is less than \$1,000, it can be approved by the user's immediate manager.

To continue working on the approval, follow these steps:

1. With the flow open, expand the **If no** condition branch. This is the condition branch that will execute if the purchase price is \$1,000 or less. Click **Add an action**, and then select the **Create an approval Approvals** action, as shown in *Figure 10.3*:

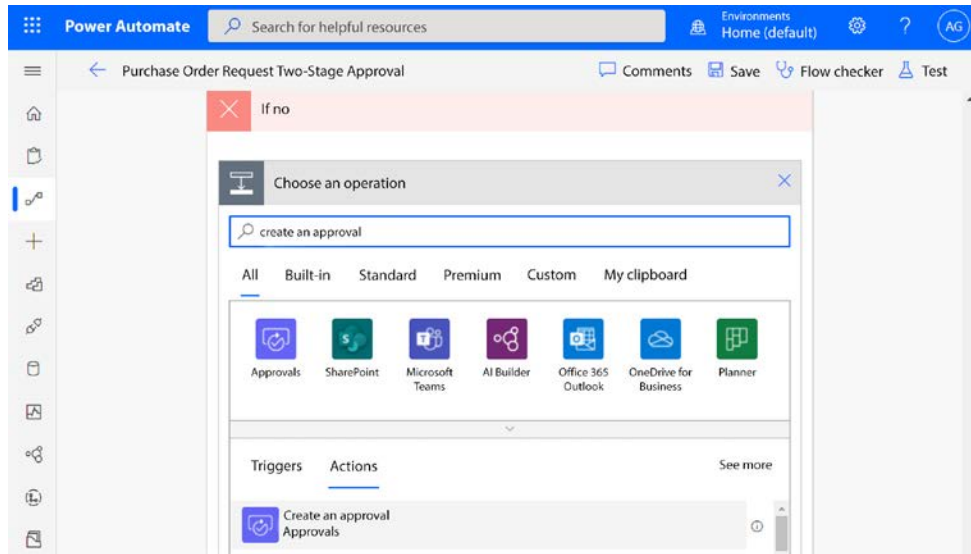


Figure 10.3: Adding the Create an approval action

2. For **Approval type**, select **Approve/Reject - First to respond**.
3. For the **Title** value, include the dynamic content token for **Title** under the **When an item is created** section, along with any other descriptive text if desired.
4. For **Assigned to**, select the **Mail** dynamic content token under the **Get manager** section.
5. For **Item link**, select the **Link to item** dynamic content token under the **When an item is created** section.



6. Click **Show advanced options**. In the **Requestor** box, add the **Created by Email** dynamic content token under the **When an item is created** section:

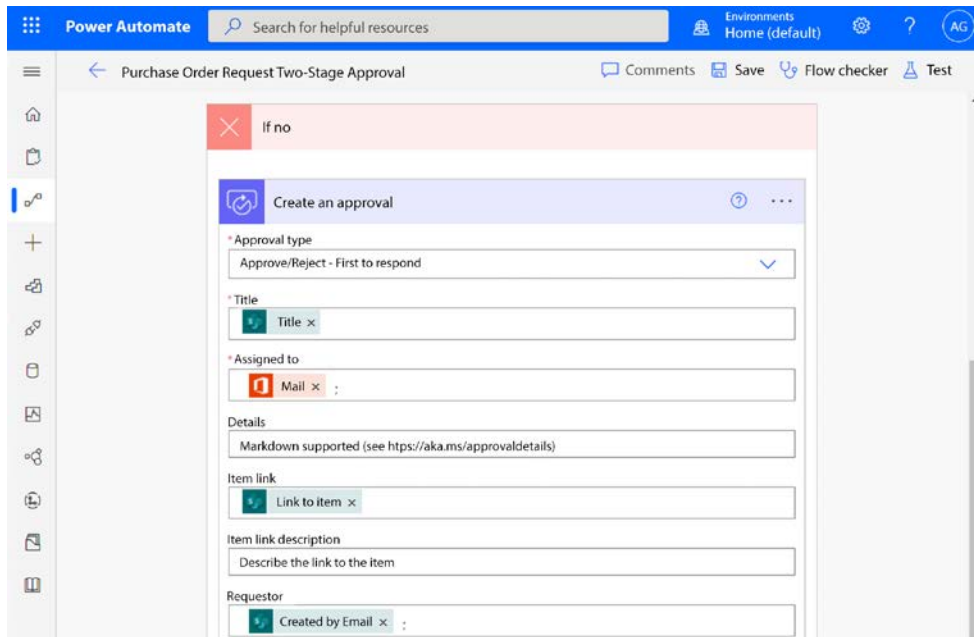


Figure 10.4: Configuring the Create an approval action under the If no branch

7. Click **Add an action**.
8. Select the **Terminate** control.
9. In the **Terminate** control **Status** box, select **Succeeded**. This will stop further steps from executing.
10. Click **Save** to save the progress of your flow.
11. Expand the **If yes** condition branch.
12. Click **Add an action**, and then select the **Update item** SharePoint action.
13. In the **Site Address** field, select the site containing the SharePoint list used for this workflow. If the site does not show up in the drop-down list, select the **Enter custom value** option and paste in the URL of the SharePoint site.
14. In the **List Name** field, select the list used for this workflow.
15. In the **Id** field, select the **ID** dynamic content token in the **When an item is created** section.

16. Set the **Title** field to the **Title** dynamic content token in the **When an item is created** section.
17. Set the **Purchase Amount** field to the **Purchase Amount** dynamic content token in the **When an item is created** section.
18. Ensure the **Pre-approved** dropdown is set to **No**.
19. Ensure the **Approved** dropdown is set to **No**.
20. Set the **Pre-approval required** dropdown to **Yes**. See *Figure 10.5* for an example of the completed **Update item** action:

The screenshot shows the 'Update item' action configuration in Power Automate. The action is titled 'Update item' and is part of a flow named 'Purchase Order Request Two-Stage Approval'. The configuration fields are as follows:

- Site Address:** `https://learningpowerautomate.sharepoint.com/sites/PurchaseOrder`
- List Name:** `Purchase Order Requests`
- Id:** `ID`
- Title:** `Title`
- Purchase Amount:** `Purchase Amo...`
- Comments:** (Empty text box)
- Pre-approval required:** `Yes`
- Pre-approved:** `No`
- Approved:** `No`

*Figure 10.5: Completing the Update item action*

21. Click **Add an action**, and then select the **Start and wait for an approval** Approvals action.
22. For **Approval type**, select **Approve/Reject - First to respond**.
23. For **Title**, include the dynamic content token for **Title** under the **When an item is created** section, along with any other descriptive text.
24. For **Assigned to**, select the **Mail** dynamic content token in the **Get manager (V2)** section.

25. For **Item link**, select the **Link to item** dynamic content token in the **When an item is created** section. The completed condition steps should look similar to *Figure 10.6*:

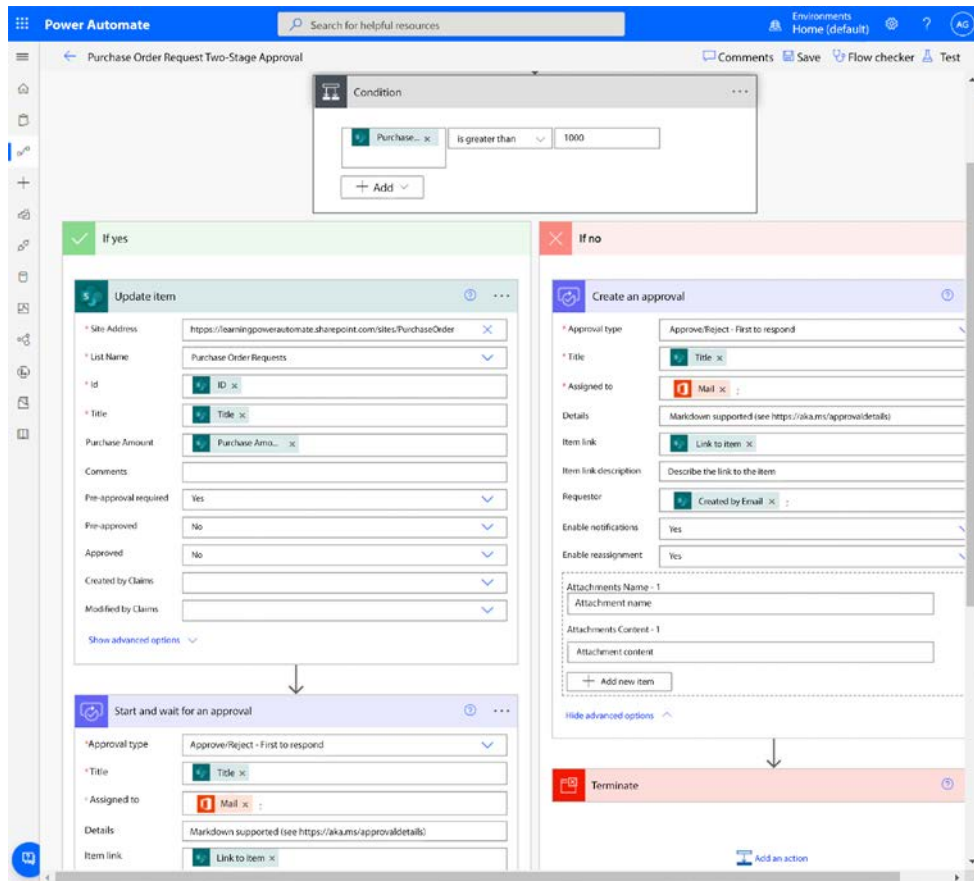


Figure 10.6: Partially completed Condition control configuration

26. Click **Save**.
27. Scroll to the bottom of the flow. Click **New step**.
28. Select **Condition control**. By default, the action will be labeled *Condition 2*. Optionally, click the ellipsis on the **Condition 2** action's title bar, and then select **Rename**. Update the name of the action to something more descriptive, such as **Evaluate Stage 1 Approval Condition**.
29. In the **Condition** control, select the **Choose a value** box on the left, and then select the **Outcome** dynamic content token in the **Start and wait for an approval** section.
30. In the **Choose a value** box on the right, enter the text **Approve**.

31. In the **If no** branch for the **Evaluate Stage 1 Approval Condition** action, click **Add an action** and select the **Send an email (V2)** Office 365 Outlook action.
32. Add the **Created by Email** dynamic content token under the **When an item is created** section to the **To** field.
33. Add a deny message to the **Subject** field. You may want to include the **Title** SharePoint dynamic content token.
34. Add a deny message to the **Body** field. You may want to use the **Title** SharePoint dynamic content token as well as the **Display Name** dynamic content token from the **Get manager (V2)** action:

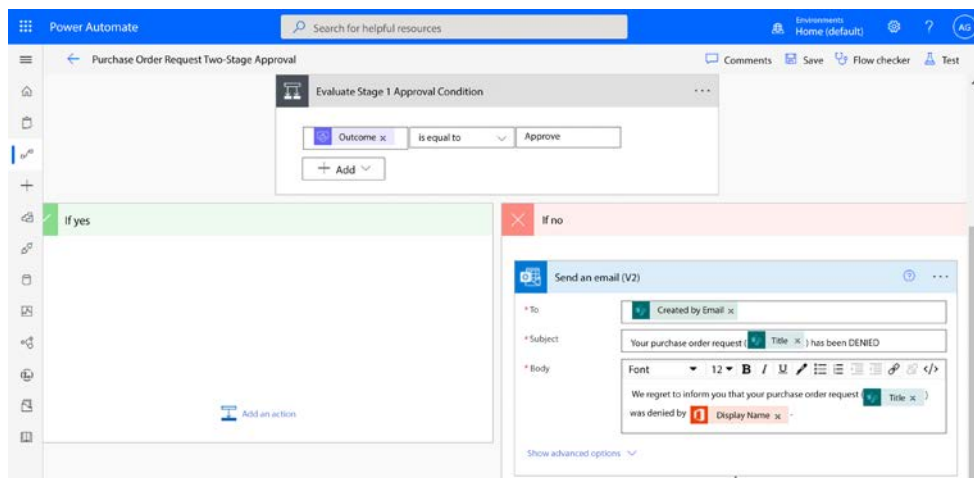


Figure 10.7: Working through the Evaluate Stage 1 Approval Condition

35. Click **Add an action**, and then select the **Terminate** control.
36. Select **Succeeded** as the **Status**.
37. Click **Save**.

At this point, the first stage of the approval process is complete. For purchase orders involving sums of \$1,000 or less, the immediate manager can approve or reject the request. For purchase orders over \$1,000, the SharePoint item for the approval is updated to a *pending* approval state, and a new approval process is started.

As part of the approval process, the flow sends a notification to the requester's manager. If it is denied, the approval ends; the final step of the process is to send an email to the requester indicating the approval has been denied. If the outcome is an approval, however, the flow will start a new branch to kick off the second stage.



### Renaming steps

By default, a step is labeled with the name of the action. However, in long or complex flows with several conditions or branches, it may be helpful to rename particular steps to help document where you are in the process. To rename a step, click on the ellipsis (...) in the title bar of the step and select **Rename**. Each step in a flow must have a unique name value. While you can rename a step at any time, you may need to go back and double-check to ensure your dynamic content tokens reference the correct values. It's recommended to rename the actions as you create them instead of going back to update them later.

Now, let's move on to creating the next stage of the approval.

## Creating the second-stage approval

In this section of the approval process, the request is sent to the second-level or second-stage approver. The second-stage approver will also have the ability to approve or deny the request.

To complete the flow, follow these steps:

1. With the flow open, select the **If yes** branch of the second **Condition control, Evaluate Stage 1 Approval Condition**, and then click **Add an action**.
2. Select the **Update item** SharePoint action.
3. In the **Site Address** field, select the site containing the SharePoint list used for this workflow.
4. In the **List Name** field, select the list used for this workflow.
5. In the **Id** field, select the **ID** dynamic content token under the **When an item is created** section.
6. Set the **Title** field to the **Title** dynamic content token under the **When an item is created** section.
7. Set the **Purchase Amount** field to the **Purchase Amount** dynamic content token in the **When an item is created** section.
8. Set the **Pre-approved** dropdown to **Yes**.
9. Ensure the **Approved** dropdown is set to **No**.
10. Ensure the **Pre-approval required** dropdown is set to **Yes**.
11. Click **Add an action**.

12. Select the **Start and wait for an approval** Approvals action. By default, the step will be titled **Start and wait for an approval 2**. Optionally, rename the Approval action to something more meaningful.
13. For the **Approval type** field, select **Approve/Reject - First to respond**.
14. For the **Title** field, select the **Title** dynamic content token from the **When an item is created** section.
15. For the **Assigned to** field, enter the **Finance Manager** email address.



While we're choosing a static email address here for the secondary approver, you could also use the **Get manager (V2)** action in conjunction with the pre-approver's address and use that as the next-level approver.

16. Review the flow to ensure it is structured similarly to the example in *Figure 10.8*:

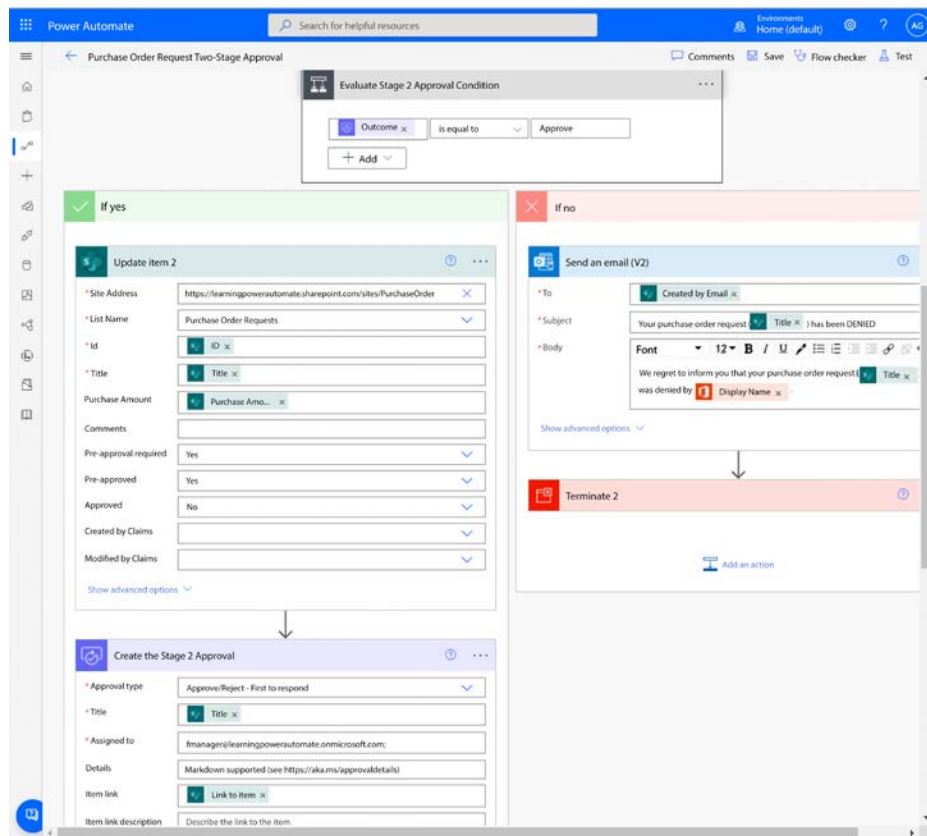


Figure 10.8: Quickly checking the structure of the flow up to this point

17. Click **Save**.

You've now completed the second-stage approval section. Next, we'll move on to completing the flow.

## Completing the flow

In this final section of the flow creation, we'll follow a process similar to the pre-approval: using a condition to check the value of **Outcome** and then updating the SharePoint list item tied to the purchase order request accordingly.

To complete this flow, follow these steps:

1. With the flow open, scroll to the bottom of the flow and click **New step**.
2. Add a **Condition** control. Optionally, rename the new **Condition** step to **Evaluate Stage 2 Approval Condition**.
3. For the **Choose a value** box on the left side of the condition, select the **Outcome** value in the **Start and wait for an approval** section for the final approval (either **Start and wait for an approval 2** or your custom name).
4. Select **is equal to** as the evaluation method for the condition.
5. Set the **Choose a value** box on the right side of the condition control to **Approve**.
6. In the **If no** condition branch, click **Add an action**.
7. Select the **Send an Email (V2)** Office 365 Outlook action.
8. In the **To** field, add the **Created by Email** dynamic content value under the **When an item is created** section.
9. In the **Subject** field, add a message. Optionally, add the **Title** dynamic content value.
10. In the **Body** field, add a message body. Optionally, add the **Title** dynamic content value and any other values (such as the **Approver** name).
11. Click **Add an action** and select the **Update item** dynamic content value in the **When an item is created** section.
12. Configure the **Site Address**, **List Name**, **Id**, **Title**, and **Purchase Amount** values using the dynamic content value tokens from the **When an item is created** section.
13. Set the **Pre-approved** field to **No**.
14. Set the **Approved** field to **No**.
15. Set the **Pre-approval required** field to **Yes**.
16. Click **Add an action** and then select the **Terminate** control.

17. Set the **Status** value to **Succeeded**.
18. Review the **If no** branch to ensure it is structured like *Figure 10.9*:

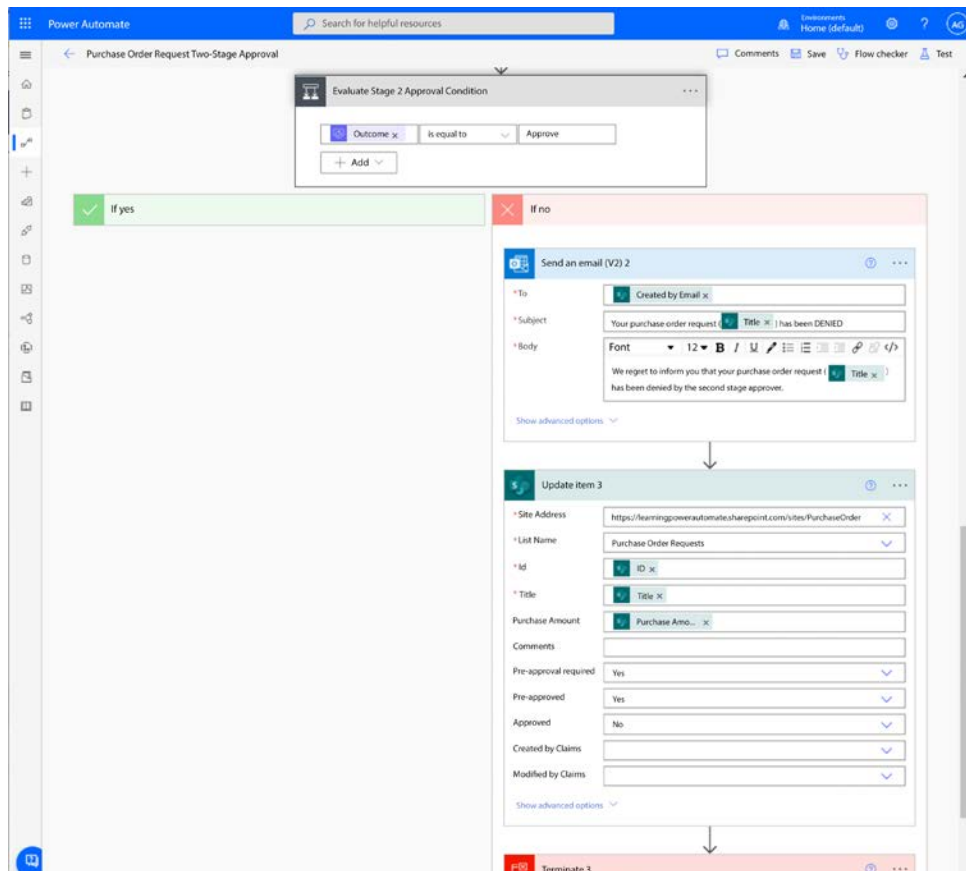


Figure 10.9: Configuring the final If no branch

19. Select the final **If yes** condition branch, and then select **Add an action**.
20. Select the **Update item** SharePoint action.
21. Configure the **Site Address**, **List Name**, **Id**, **Title**, and **Purchase Amount** values using the dynamic content value tokens from the **When an item is created** section.
22. Set the **Pre-approved** field to **Yes**.
23. Set the **Approved** field to **Yes**.
24. Set the **Pre-approval required** field to **Yes**.
25. Select the **Send an Email (V2)** Office 365 Outlook action.



26. In the **To** field, add the **Created by Email** dynamic content value under the **When an item is created** section.
27. In the **Subject** field, add a message. Optionally, add the **Title** dynamic content value.
28. In the **Body** field, add a message body. Optionally, add the **Title** dynamic content value and any other values (such as the **Approver** name).
29. Click **Add an action** and select the **Terminate** control.
30. Set the **Status** field to **Succeeded**.
31. Review the final **If yes** branch to ensure that it is structured similarly to *Figure 10.10*:

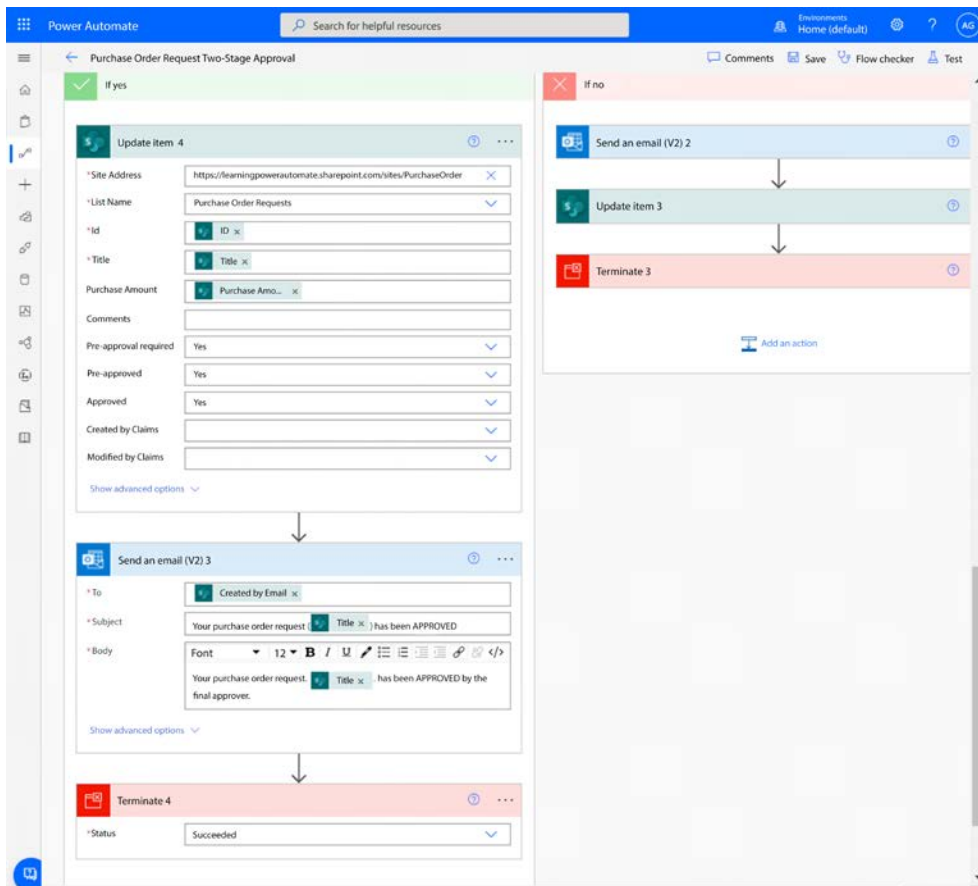


Figure 10.10: Reviewing the final If yes branch

32. Click **Save**.

The approval has been successfully created! Since this is an especially long and complex flow, you may want to review the items and update any descriptions or names of steps to clarify them for others who may look at the flow (or for yourself if you need to perform troubleshooting).

When finished, it's time to test!

### Testing the flow

To test this flow, navigate to the SharePoint site containing the list being monitored and add values. In order to verify that it works as designed, you may need to run several tests, including requests over and under \$1,000, and then approve and deny them at either the first or second stage.

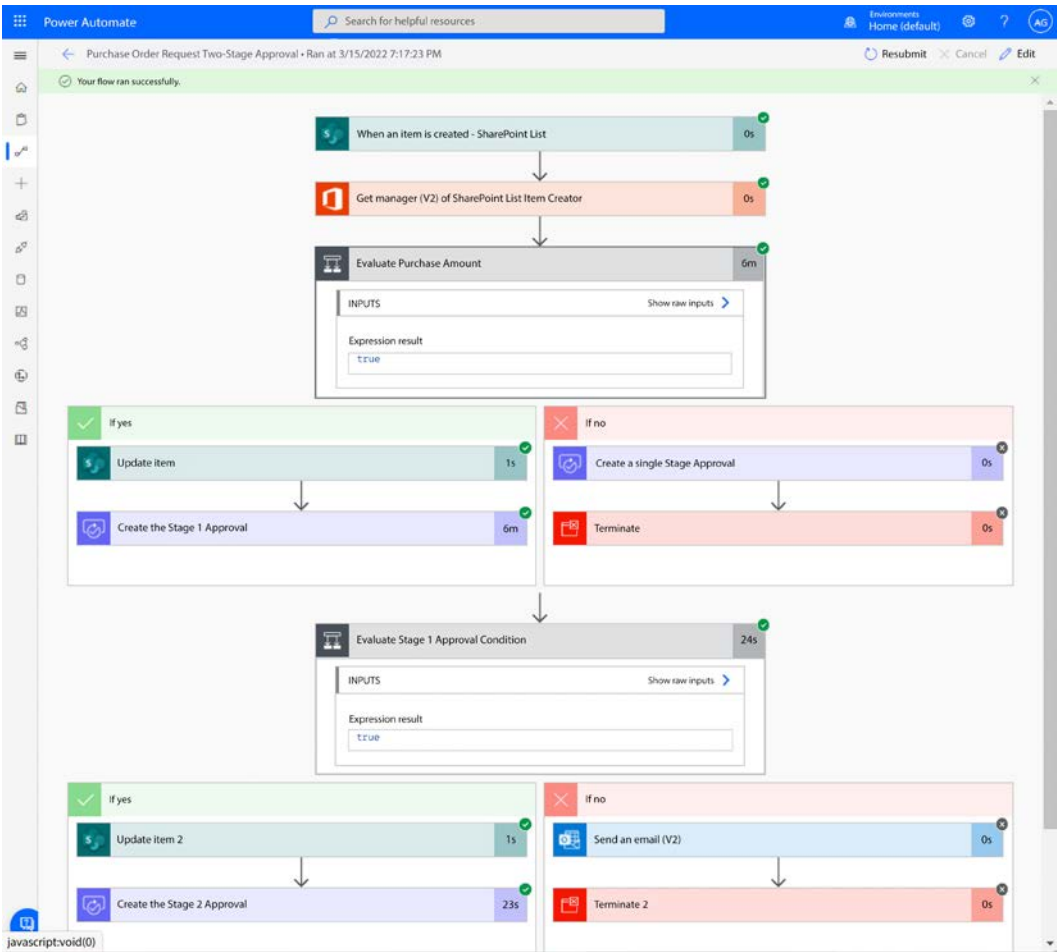


Figure 10.11: Reviewing the flow run history

If any of the steps error, review the flow steps to ensure you're using the appropriate dynamic content tokens and that condition evaluation statements are correct.

As you become more familiar and confident with Power Automate concepts, you'll discover new ways to customize the outputs or steps, including using complex expressions, additional formatting, and combining dynamic content values to produce rich output.

## Adding parallel branches

For approval processes that are either long-running or involve multiple users or interactions, you may want to send periodic email reminders to users. One way to do this is by populating variables at each approval stage and checking whether they're complete using a parallel approval branch. You'll do this with five components:

- The **Initialize variable** action will be used to configure a variable (such as `FirstApprovalDone`) that you'll check throughout the approval. The variable will be initialized to **false**, and then only set to **true** once that approver approves the request.



With some programming languages, you can save the results of an action or command to a variable on the fly. With Power Automate, you must declare variables before you use them.

- The **Do until** control will be used to create a loop that periodically checks to see whether the variable (`FirstApprovalDone`, in the example) has changed from **false** to **true**.
- The **Delay** action will allow you to delay the sending of the email by some period of time (to give the approver a chance to respond before getting a notification).
- The **Send an email (V2)** action, which will send the reminder email to the approver.
- The **Set variable** action will be used to update the variable (`FirstApprovalDone`) to signify that the approval has been completed.

You can use the following steps to add reminder emails:

1. Using the Power Automate web portal interface (<https://flow.microsoft.com>), edit the two-stage approval.
2. Navigate to the **Get manager (V2)** step, hover over the arrow below it, select the + icon, and then select **Add an action**:

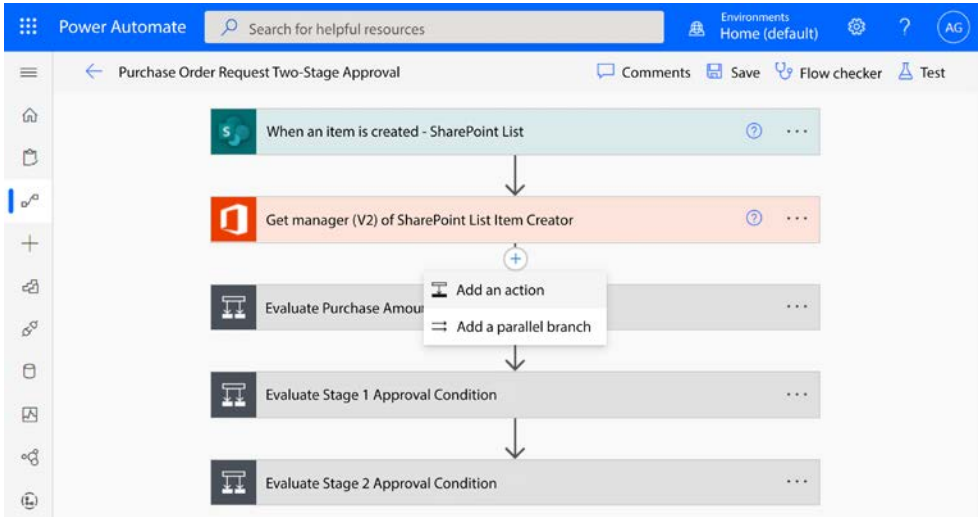


Figure 10.12: Inserting a new action between steps

3. Select the **Initialize variable** action:

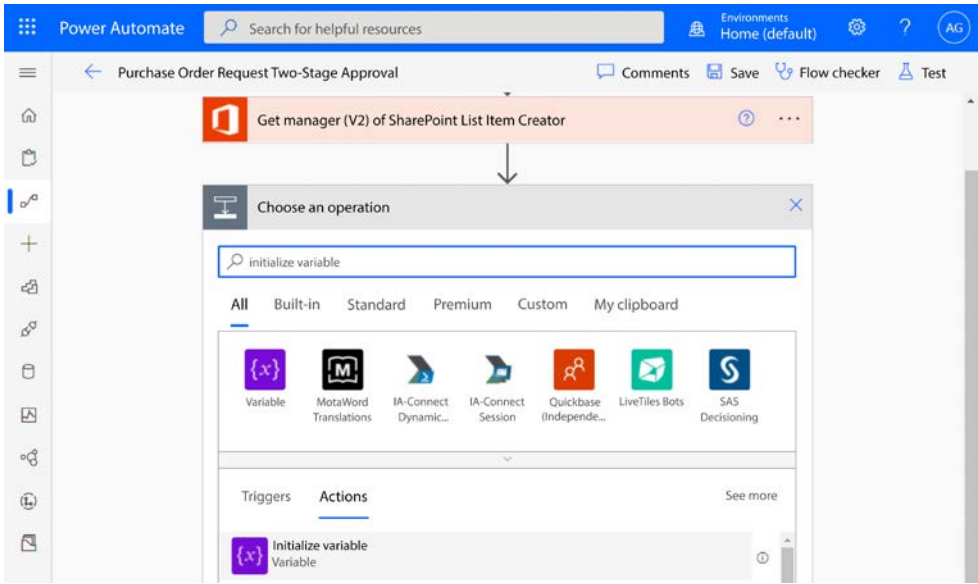


Figure 10.13: Selecting the Initialize variable action

Name the variable something that will help you keep track of which approval you’re working with. Set the type to **Boolean** and enter a starting value of **false**.

4. In this case, you're going to use this variable to track whether the first approval has been done, so name it **FirstApprovalDone**:

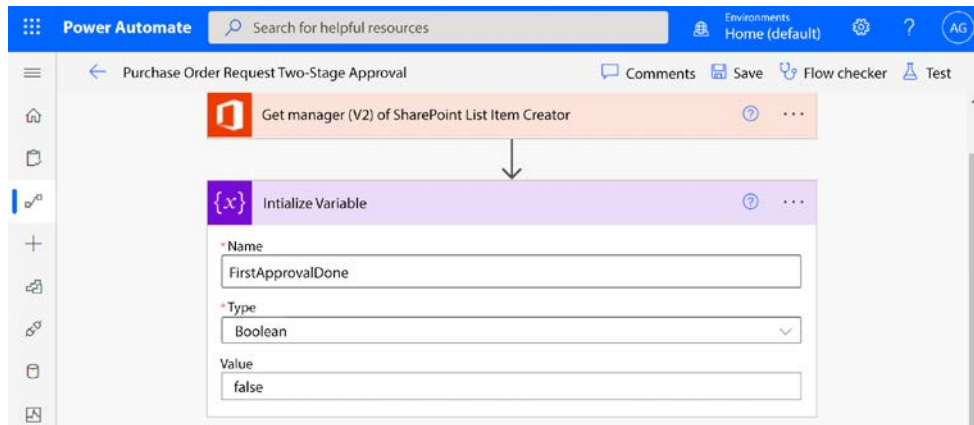


Figure 10.14: Initializing the *FirstApprovalDone* variable

5. Next, expand the condition where the first approval is created (it should be the second condition control step).
6. In the **If yes** branch, select the + icon between the **Update item** and **Start and wait for an approval** actions, and then select **Add a parallel branch**:

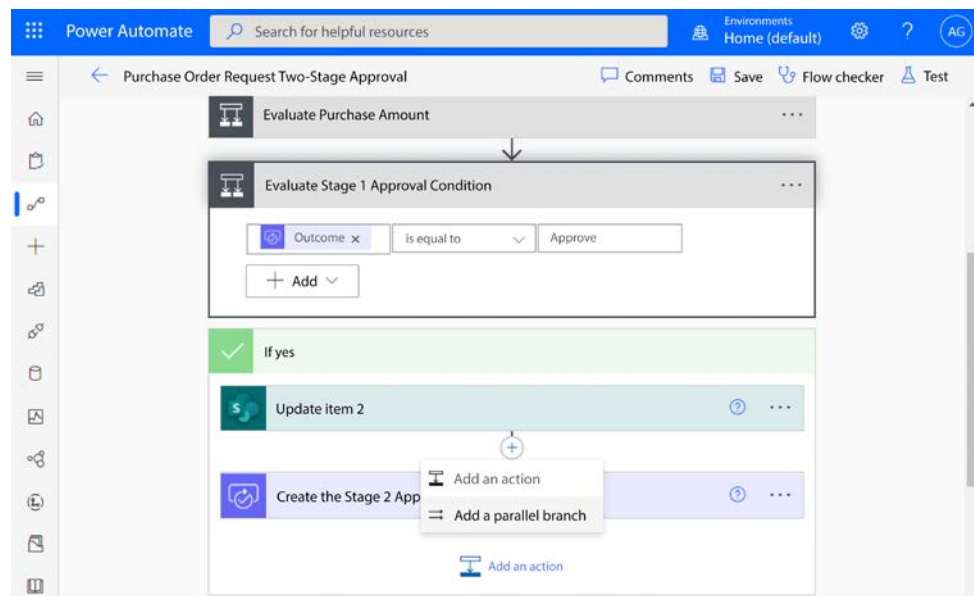


Figure 10.15: Adding a parallel branch

7. In the new branch, select the **Delay Schedule** action, as shown in *Figure 10.16*:

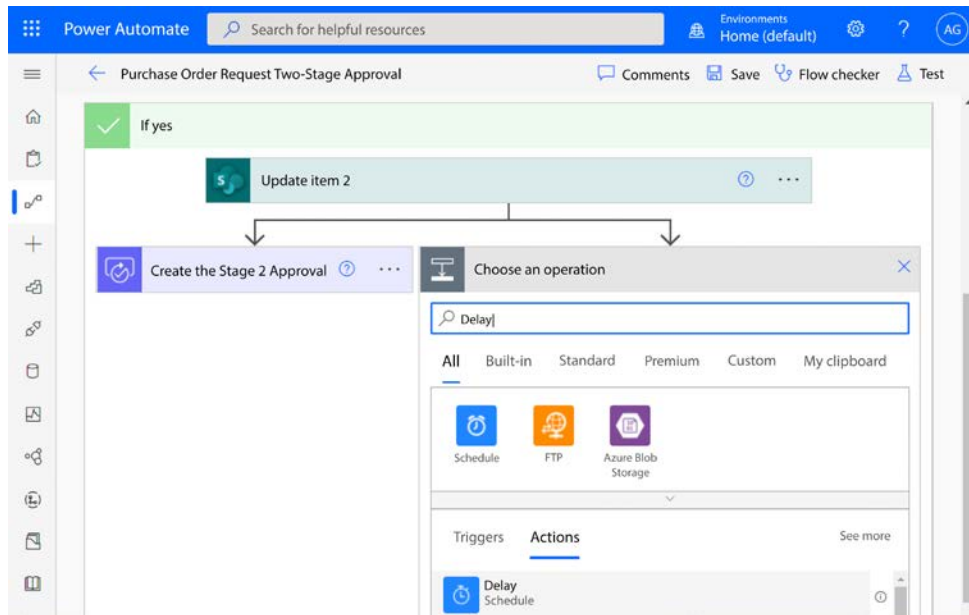


Figure 10.16: Adding the Delay action

8. Enter a number for **Count** and a time for **Unit**, which will be used as a delay for sending the first reminder email. In this example, selecting 1 as the **Count** and Day as the **Unit** means that the first reminder email won't go out until after 1 day has elapsed:

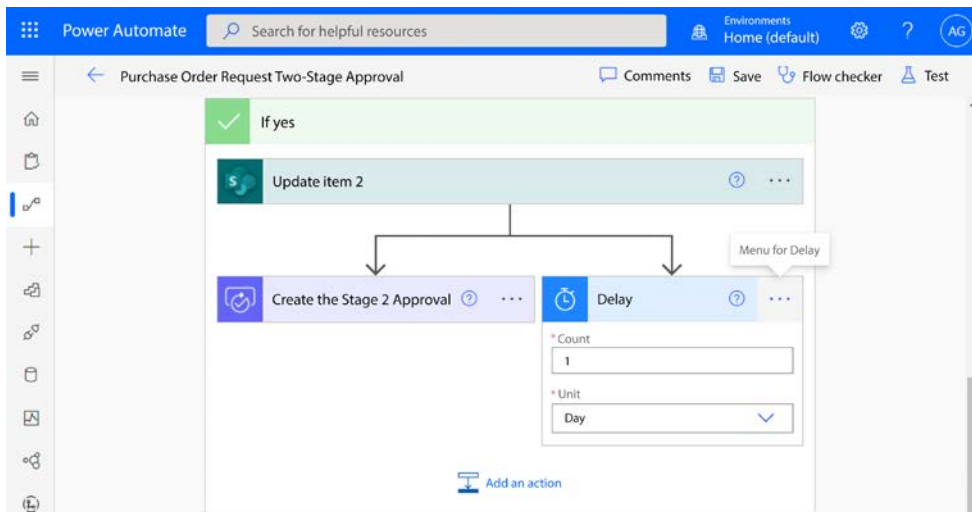


Figure 10.17: Configuring the Delay action

- Under the **Delay** action, click + to insert a step. Select **Add an action**:

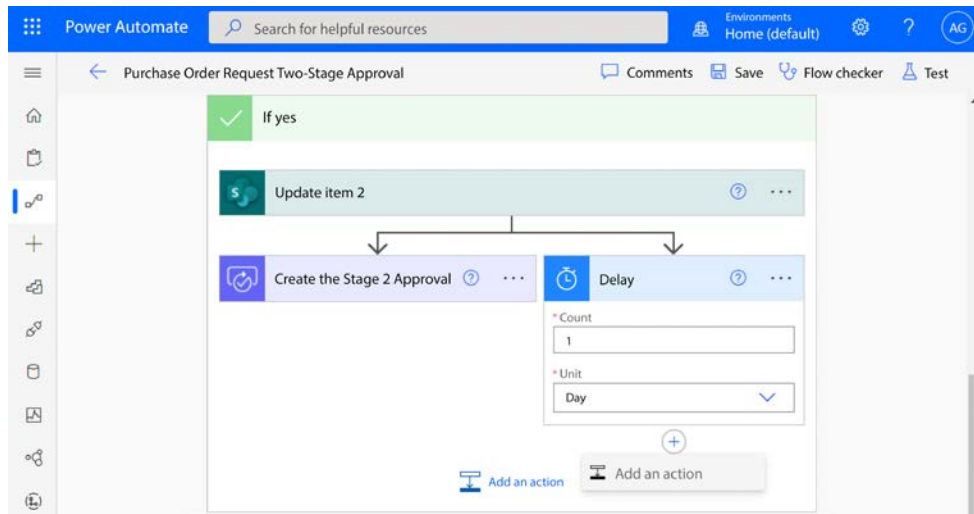


Figure 10.18: Adding an action after the Delay action

- Select the **Do until** control:

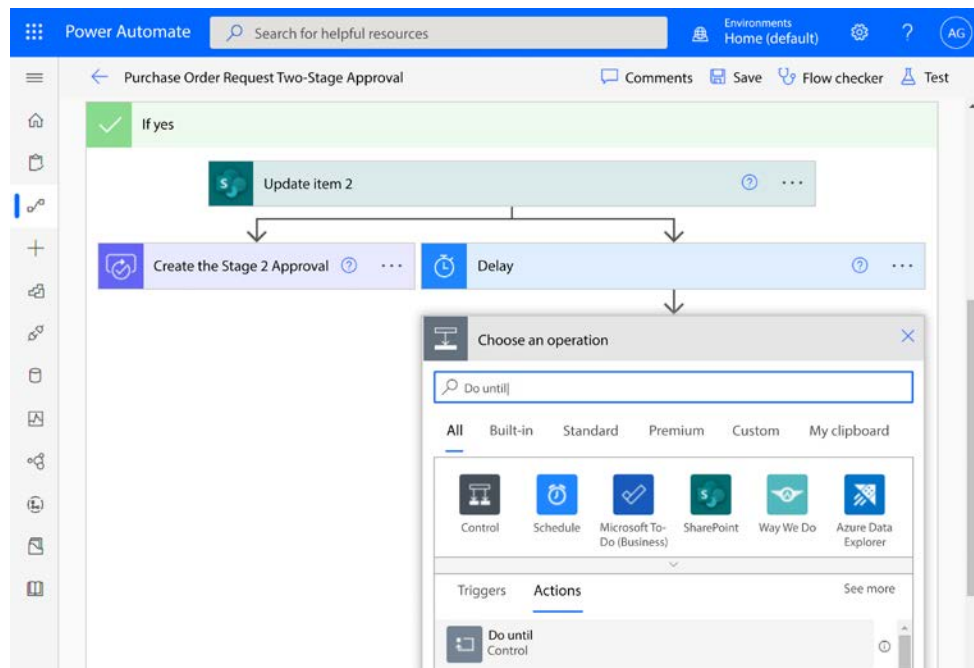


Figure 10.19: Adding the Do until control

11. In the **Choose a value** box on the left side of the **Do until** action, add the dynamic content token for **FirstApprovalDone**.
12. In the **Choose a value** box on the right side of the **Do until** action, select the expression for **true**:

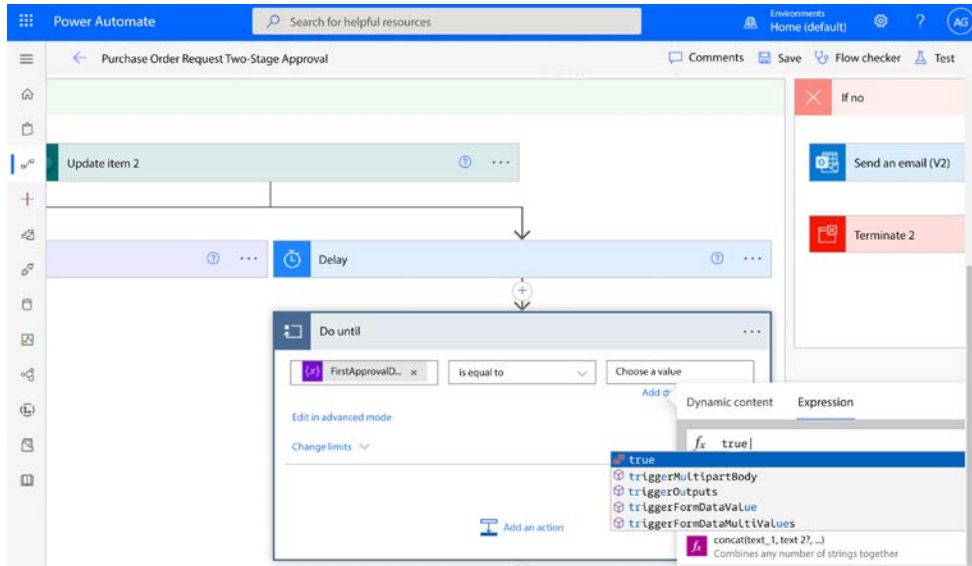


Figure 10.20: Adding the Boolean true expression to the evaluation



You can use the browser zoom feature to cause the older **Dynamic content** and **Expression** picker to display.



- 13. Click **Add an action** inside the **Do until** action card:

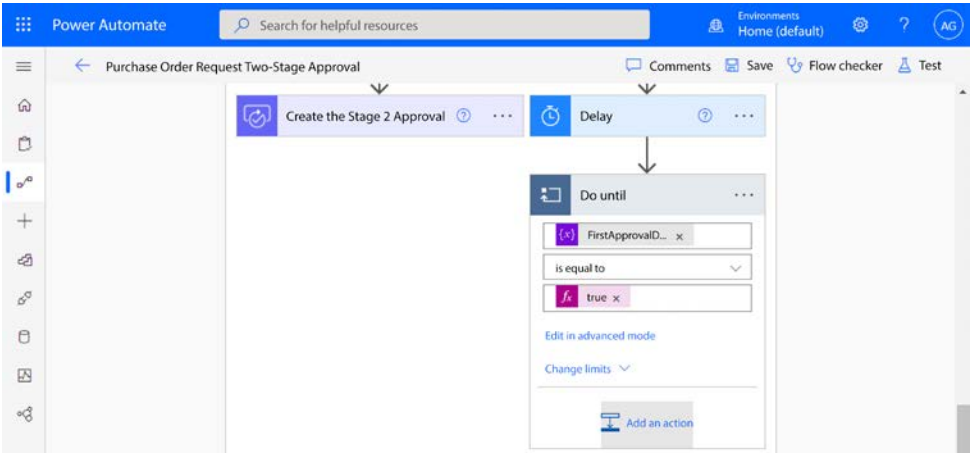


Figure 10.21: Adding an action in the Do until action

- 14. Select the **Send an email (V2)** Outlook action.
- 15. Fill out the email using the **Mail** dynamic content token value from the **Get Manager (V2)** action. Enter a subject and an email body:

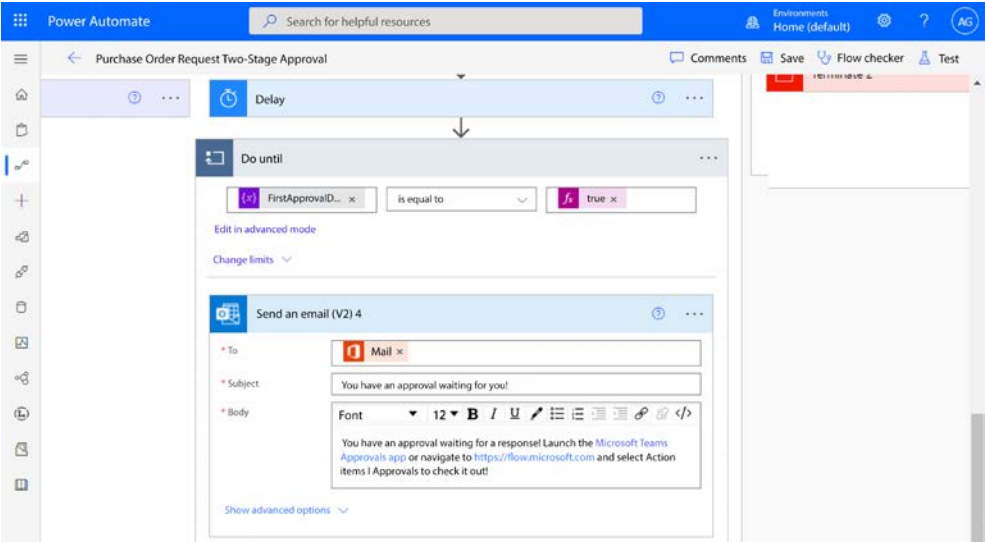


Figure 10.22: Configuring the reminder email

16. Select **Add an action** inside the **Do until** action card.
17. Select the **Delay** schedule action. As before, enter a **Count** value and select a **Unit** measurement. This will determine *how often* reminder emails get sent.

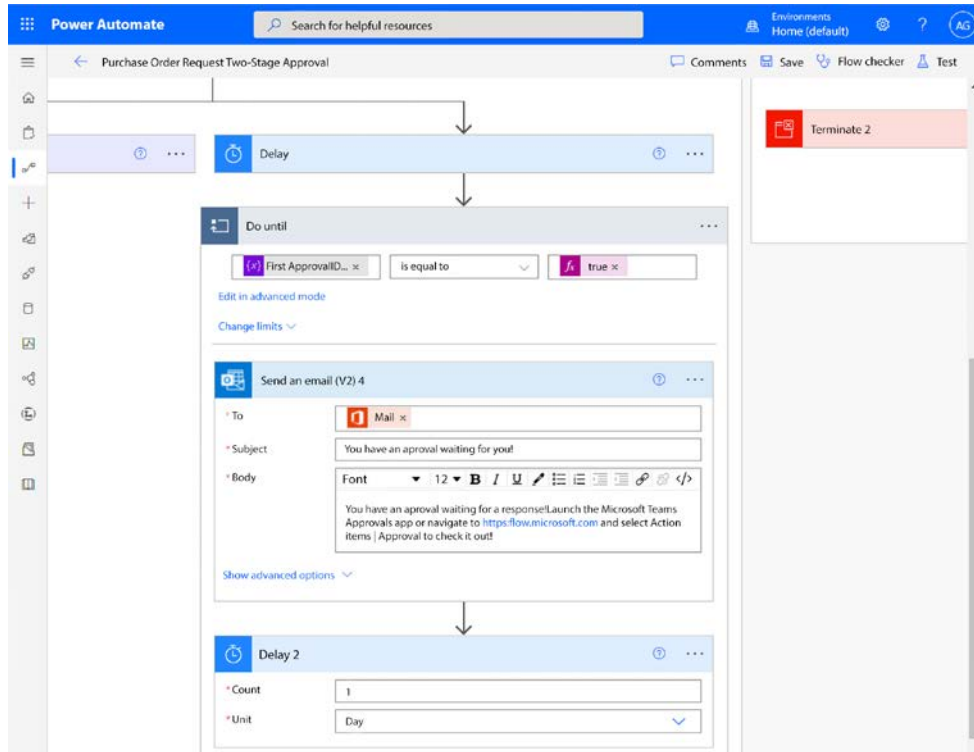


Figure 10.23: Configuring the reminder interval

18. Minimize the **Do until** card. Select the + icon to add a new step under the first branch containing **Start and wait for an approval**:

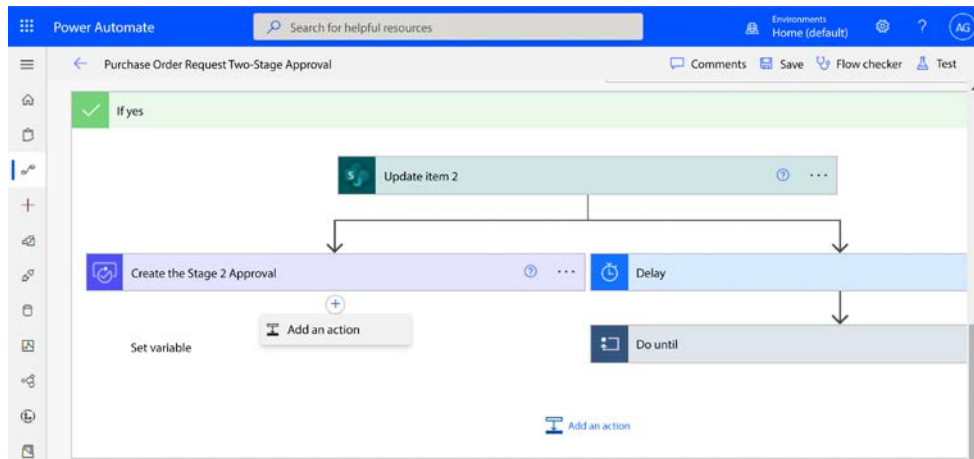


Figure 10.24: Adding an action after the Create the Stage 2 Approval action

19. Add the **Set variable** action.
20. Select the **FirstApprovalDone** variable from the **Name** dropdown and then add the **true** expression:

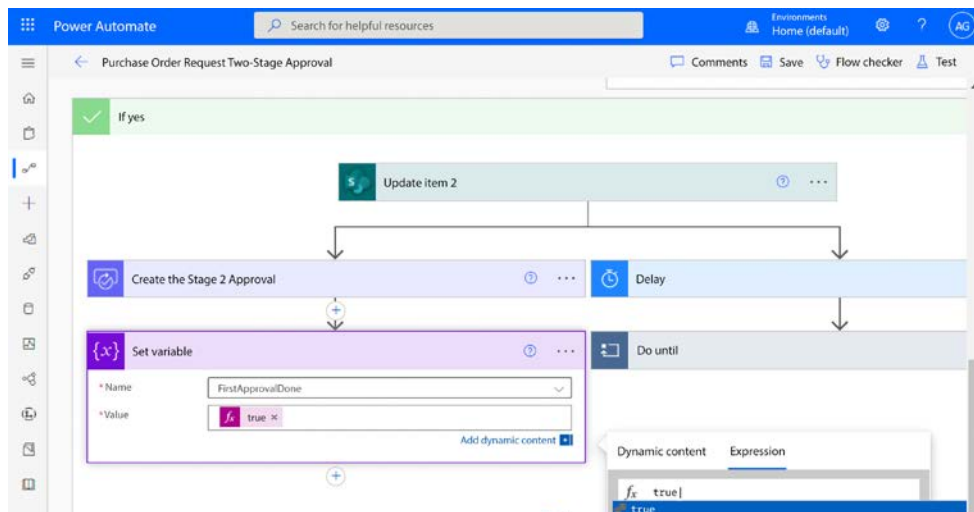


Figure 10.25: Configuring the Set variable action

21. Click **Save**.

Depending on how many reminder branches you want to instantiate, you can repeat this process for each approval throughout the flow.

## Summary

In this chapter, you learned how to start creating approval flows that require multiple stages or approvals to be successful. Sequential approvals require a series of approvals to be completed in sequence, while parallel approvals can involve multiple branches of approval actions happening independently.

These types of complex approval request flows are frequently used in medium- and large-sized organizations and reflect real-world scenarios. Becoming familiar with these types of approvals will help you create flows that more accurately meet your organization's needs in the future.

In addition to multi-stage approvals, you also worked with parallel branches and used delays, loops, and variables to create a framework for sending reminder emails on a regular schedule.

In the next chapter, we'll look at integrating the approvals process with Microsoft Teams.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>





# 11

## Posting Approvals to Teams

Microsoft has positioned Teams as the “hub for teamwork” (<https://docs.microsoft.com/en-us/microsoftteams/teams-overview>, see *Teamwork*). With that in mind, we’re going to apply the approval flow skills you acquired in *Chapter 9, Getting Started with Approvals*, and use them to build an approval experience in Microsoft Teams. You’ll be able to use these skills to help streamline the approvals process for members of your organization that are using Teams as a platform.

In this chapter, we’re going to cover the following topics:

- Configuring prerequisites
- Configuring an approval flow to use Teams
- Testing the flow

By the end of this chapter, you’ll be able to integrate a Power Automate approval flow with Microsoft Teams.

Let’s go!

### Understanding the flow

Before we configure an approval process, you’ll need to make sure that all of the components that support the process exist. In the example flow, we’re going to configure a **Purchase Order** approval that begins when a document is posted to a SharePoint site. The approval will show up in Teams. If the purchase requisition is approved, the document will be moved to the Approved folder. If it’s rejected, it will be moved to the Rejected folder. The requester should receive an email at the end telling them the status.

This complex flow is going to use several familiar components, including the following:

- SharePoint **When a file is created in a folder** trigger
- Office 365 Users **Get user profile (V2)** action
- Office 365 Users **Get manager (V2)** action
- Approvals **Create an approval (V2)** action
- Conditions

In this chapter, we're also introducing the **Microsoft Teams connector**. With the Microsoft Teams connector, you'll be able to perform actions such as these:

- **Post your own adaptive card as the Flow bot to a user:** Adaptive cards are a new way to present content directly in a Microsoft Teams conversation or experience. Adaptive cards can contain components such as graphs, images, and formatted text.
- **Post a message as the Flow bot to a user:** This action will allow Power Automate and the Flow bot to communicate with the requester of an approval.

These new actions will allow Power Automate to interact on the Microsoft Teams platform. In order to improve business productivity, the approvers should be able to respond to the approvals inside the Microsoft Teams application.

Now that you are familiar with the components that we're going to use, let's move on to configuring the prerequisites.

## Configuring prerequisites

This approval flow will require a few components and prerequisites. You're already familiar with many of them, but we'll review the exact requirements for this flow. The next two sections will describe the requirements.

### A SharePoint site

One of the main outputs of this flow is moving documents between folders on a SharePoint site, based on whether they are approved or rejected. This flow will require a SharePoint site with a document library that contains Submitted, Approved, and Rejected folders, as shown in the following screenshot:

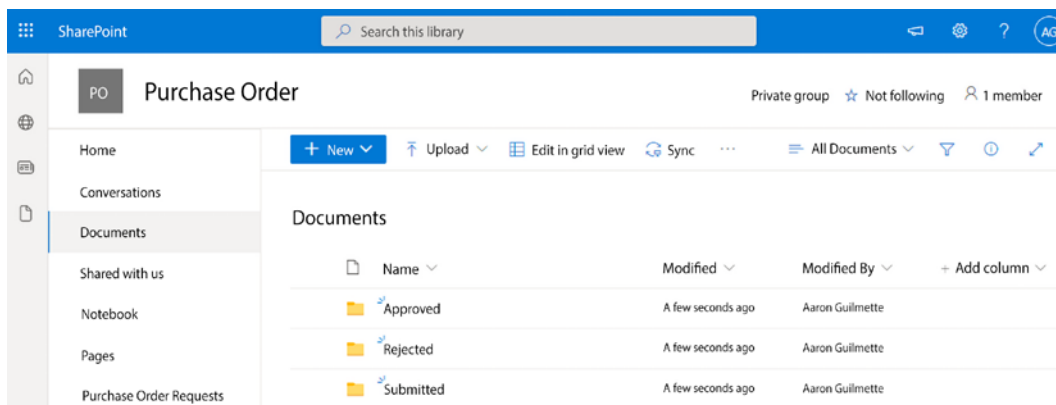


Figure 11.1: Creating folders in the default document library

For this example, we'll be using a SharePoint site called **Purchase Order**, but you can use any SharePoint site that you like.

## Power Automate app for Teams

The Power Automate app (formerly the Flow bot) for Microsoft Teams enables bidirectional interaction between Teams users and Power Automate. To install the app, you can follow these steps:

1. Launch Microsoft Teams (either the desktop application version or the web version at <https://teams.microsoft.com>) and sign in
2. On the left rail, select the ellipsis at the bottom of the apps list, and then search for Power Automate:

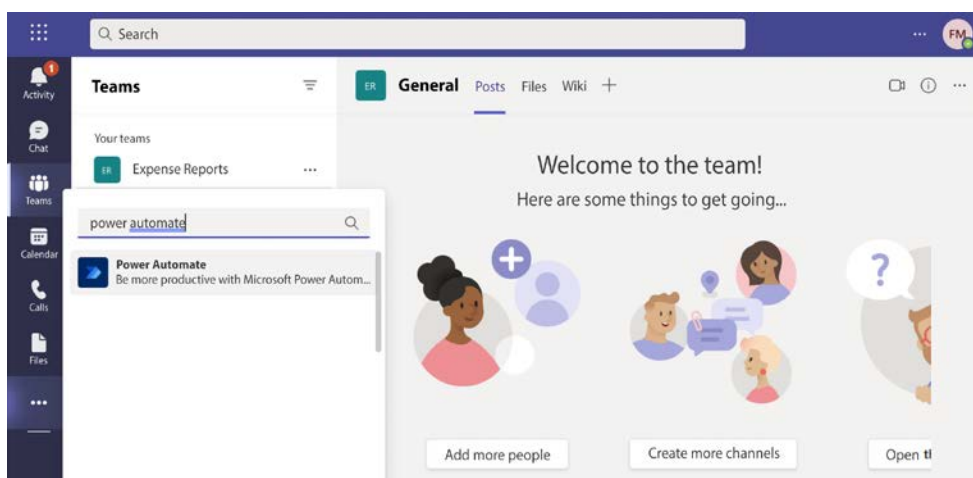


Figure 11.2: Adding the Power Automate app



3. On the app page, click **Add**:

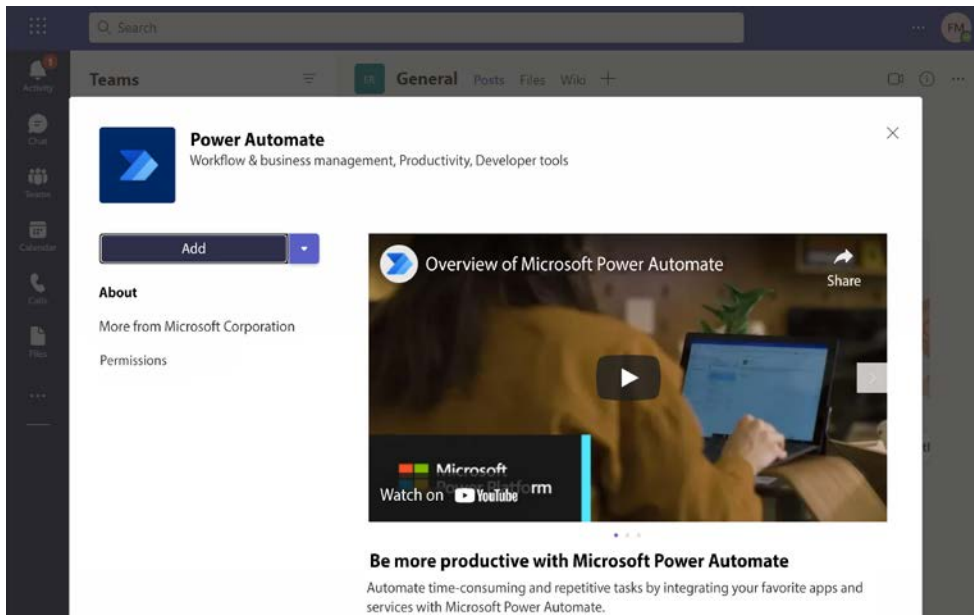



Figure 11.3: Adding the Power Automate app

Once the bot has been configured for the test users, you should be ready to begin configuring the flow.



In corporate or enterprise deployments, administrators can use Teams app setup policies to deploy or pin apps for users. For more information on Teams app setup policies, see <https://docs.microsoft.com/en-us/MicrosoftTeams/teams-app-setup-policies>.

## Configuring an approval flow to use Teams

As we mentioned in the *Understanding the flow* section, this flow is going to be initiated by a request or a file being placed on a SharePoint site and result in approval notifications being sent to both the approver and the requester inside Teams. If you want to re-familiarize yourself with some of the concepts related to files, see *Chapter 4, Copying Files*.

Since this is a longer flow, we'll complete it in stages:

- Getting the request's information
- Creating the approval

- Returning the response

Let's begin!

## Getting the requester's information

In this first section, we're going to configure the trigger for the flow to detect a new file and obtain information about the file and the requester.

To configure the flow, follow these steps:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>), select **Create**, and then select **Automated cloud flow** under the **Start from blank** section.
2. Add a name, such as **Purchase Approval**, and then choose the **When a file is created in a folder** SharePoint trigger.
3. Select the **Site Address** field and select the site you created in the prerequisites. If the site does not appear in the drop-down list, use the **Enter custom value** option and enter the URL to the SharePoint site manually.
4. Expand **Shared Documents** and then select the **Submitted** folder in the **Folder Id** field:

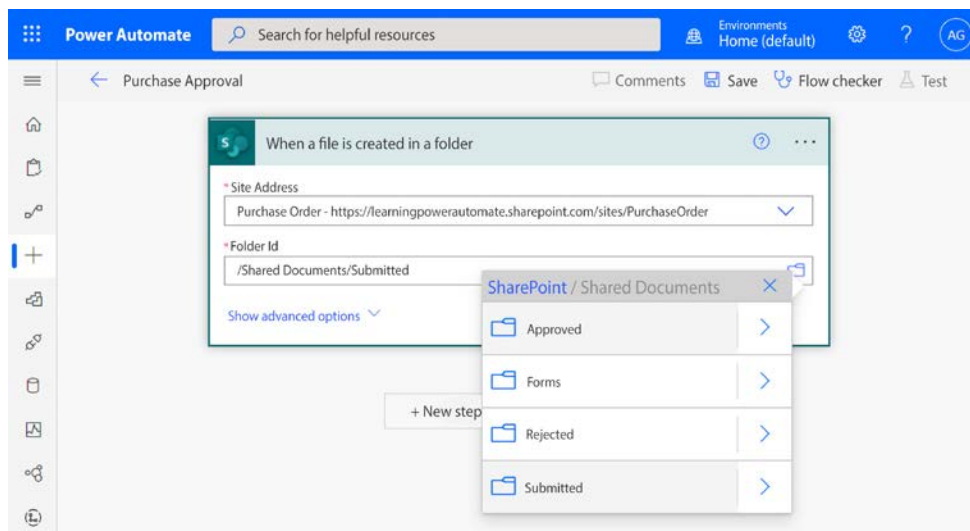
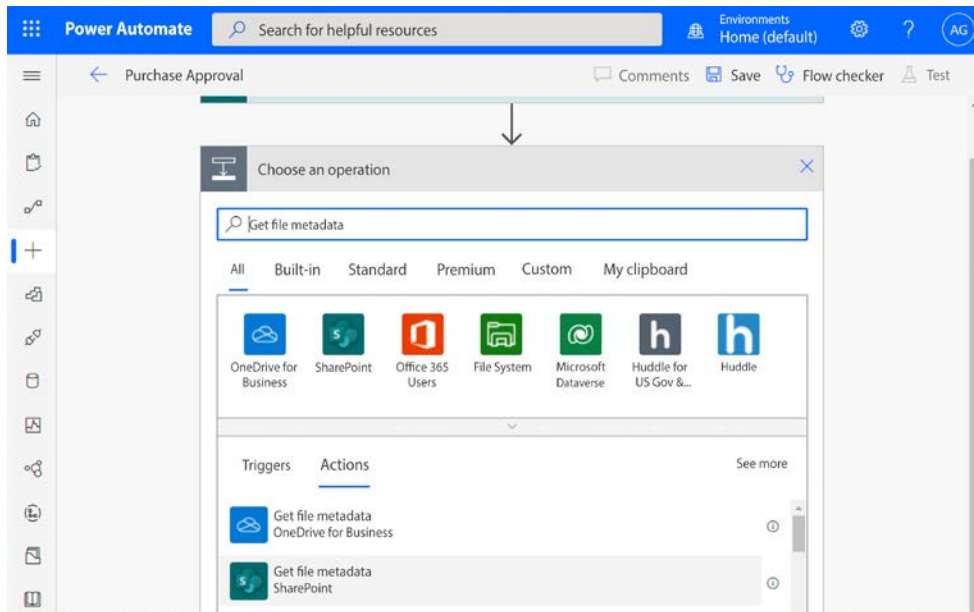


Figure 11.4: Selecting the Submitted folder

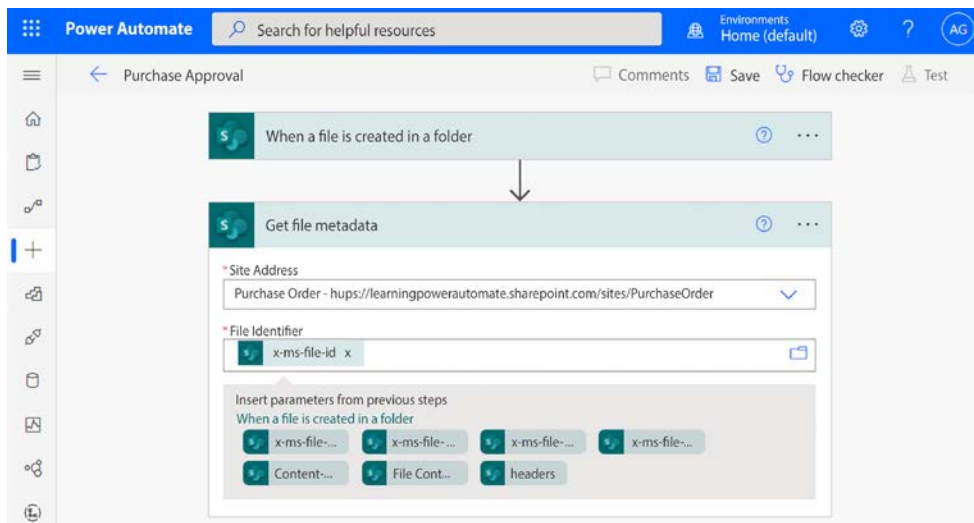
5. Click **New step**.

6. Select the **Get file metadata** SharePoint action, as highlighted in *Figure 11.5*:



*Figure 11.5: Selecting the Get file metadata SharePoint action*

7. In the **Site Address** field, select the SharePoint site containing the request. In the **File Identifier** field, select the **x-ms-file-id** dynamic content token under the **When a file is created in a folder** section:



*Figure 11.6: Configuring the Get file metadata step*

8. Click **New step**.
9. Select the **Get file properties** SharePoint action. In the **Site Address** field, select the SharePoint site containing the request. In the **Library Name** field, select the document library containing the **Submitted** folder. In the **Id** field, select the **ItemId** dynamic content token under the **Get file metadata** section.
10. Click **New step**.
11. Select the **Get user profile (V2)** Office 365 Users action. In the **User (UPN)** field, select the **Created By Email** dynamic content token under the **Get file properties** section:

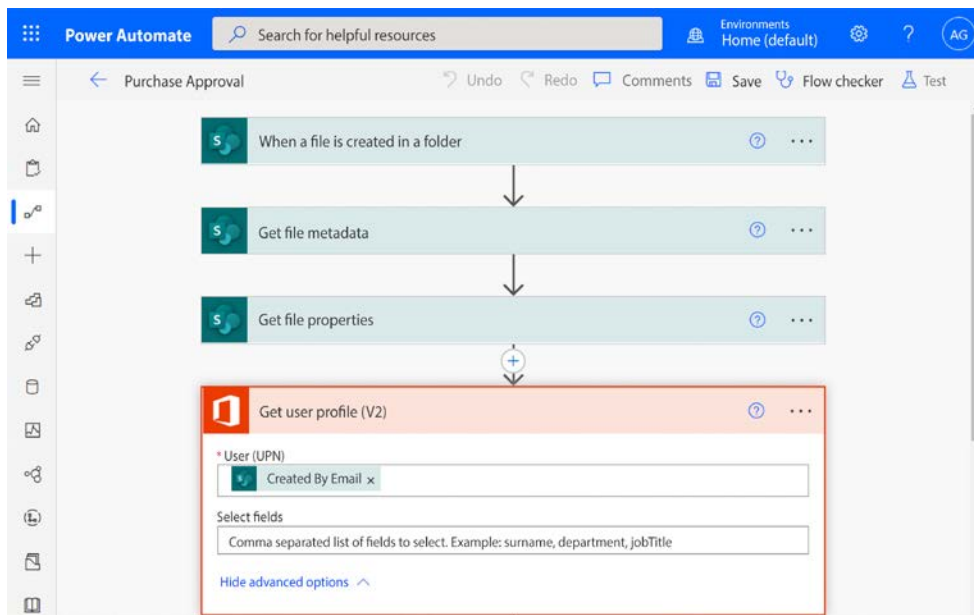


Figure 11.7: Configuring the *Get user profile (V2)* action

12. Click **New step**.
13. Select the **Get manager (V2)** Office 365 Users action.
14. In the **User (UPN)** field, select the **User Principal Name** dynamic content token under the **Get user profile (V2)** section.
15. Click **Save**.

At this point, you will have created the part of the flow that gathers information about the uploaded file and the requester.

Next, we'll configure the approval request.

## Creating the approval

In this section, you'll continue creating the approval from the previous section by adding and configuring the approval action. To complete this part of the flow, ensure that the flow from the previous section is open for editing, and follow these steps:

1. Click **New step** after the **Get manager (V2)** action:

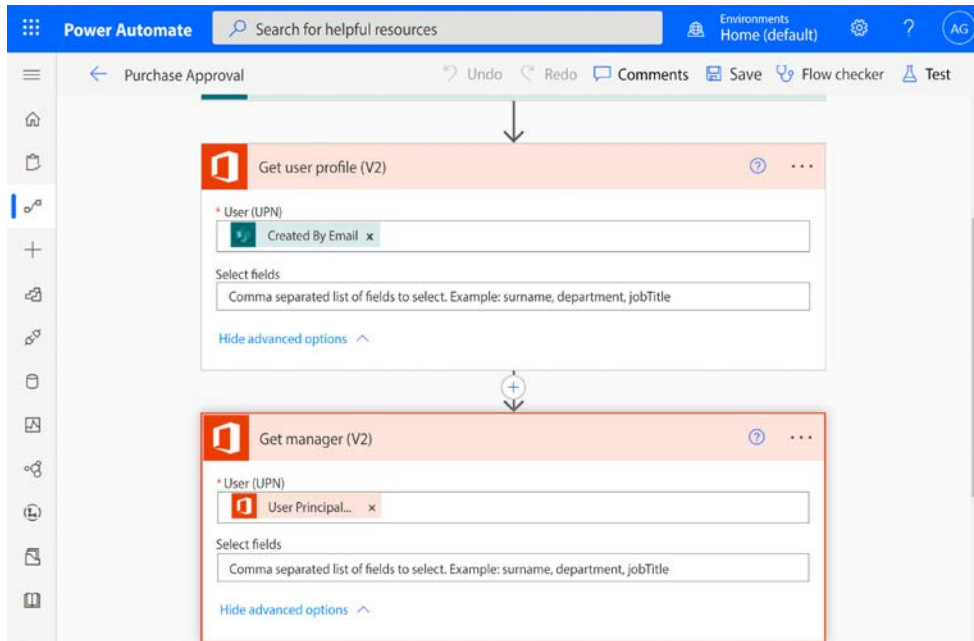


Figure 11.8: Adding a New step after the Get manager (V2) action

2. Select the **Create an approval** Approvals action.
3. Under **Approval type**, select **Approve/Reject - First to respond**.
4. Under **Title**, enter the text that uniquely identifies this request. In this example, we selected the **Title** and **Created By Displayname** dynamic content tokens from the SharePoint **Get file properties** section.

5. In the **Assigned to** field, select the dynamic content token value for **Mail** under the **Get manager** section:

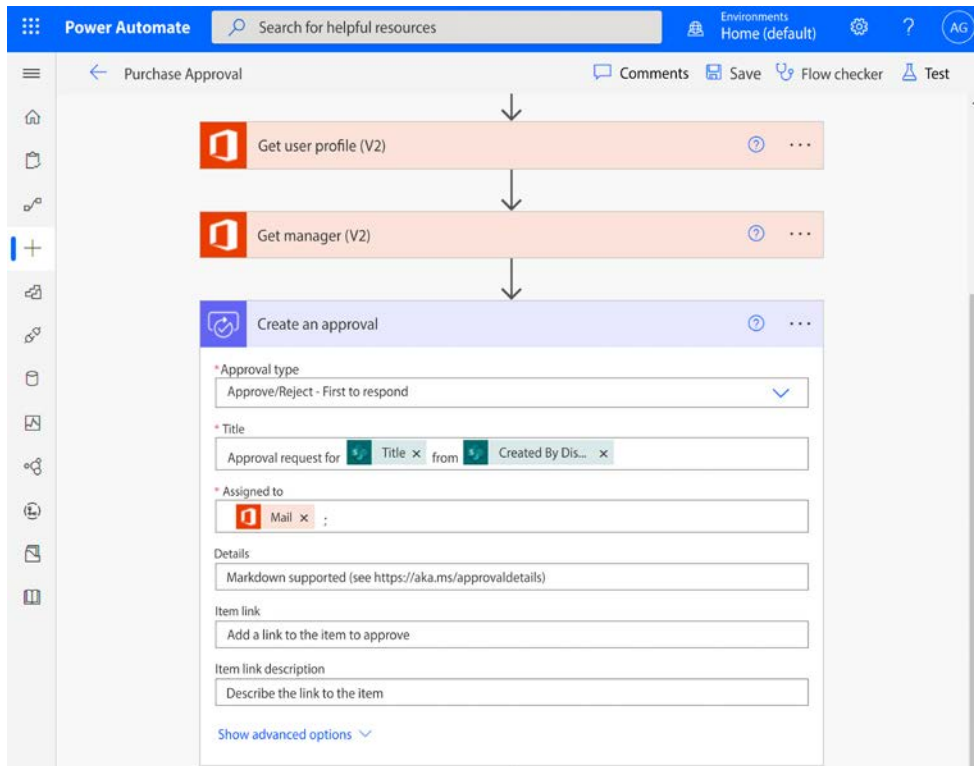


Figure 11.9: Configuring the Create an approval action

6. In the **Item link** field, you can optionally add an item, such as the **Link to item** dynamic content token under the **Get file properties** section.
7. Click **New step**.

8. Select the **Post adaptive card in a chat or channel** Microsoft Teams action:

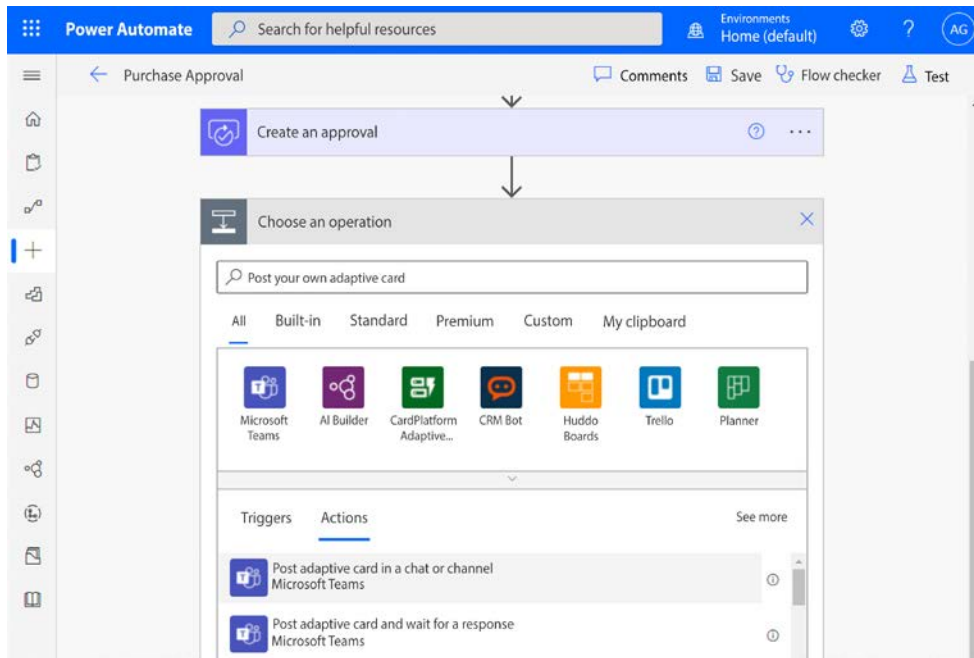


Figure 11.10: Adding an adaptive card action

9. In the **Post as** field, select **Flow bot**.
10. In the **Post in** field, select **Chat with Flow bot**. Wait for the actions on the card to dynamically update, as shown in Figure 11.11:

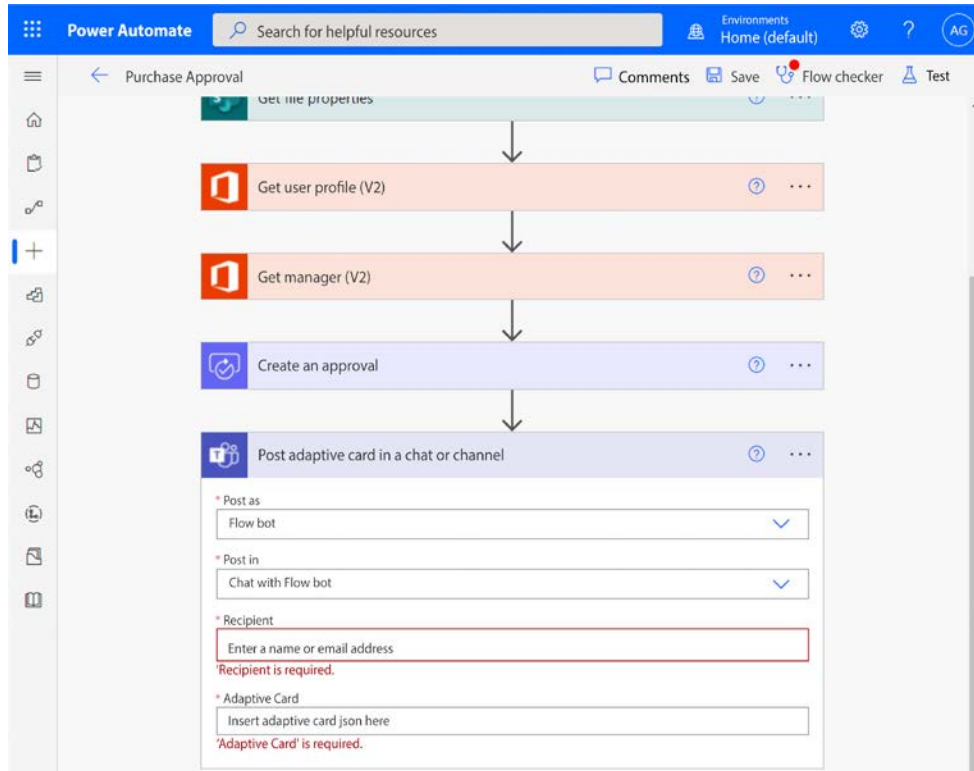


Figure 11.11: Configuring the adaptive card

11. In the **Recipient** field, select the dynamic content token for **Mail** under the **Get manager** action.



12. In the **Adaptive Card** field, select the **Teams Adaptive Card** dynamic content token under the **Create an approval** section:

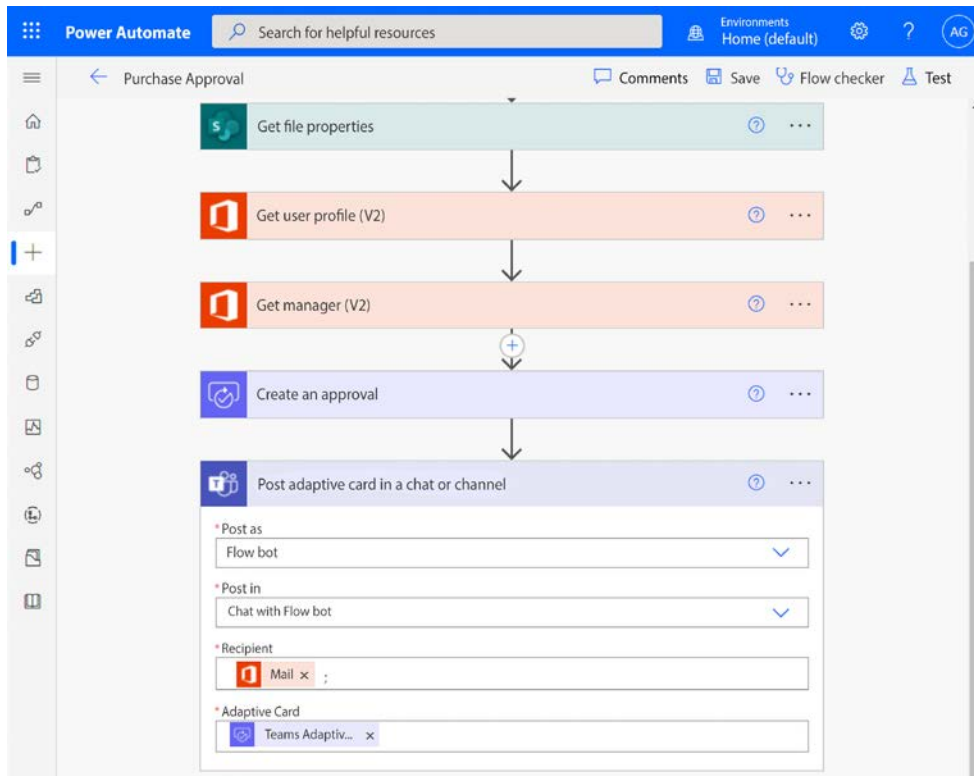


Figure 11.12: Completing the Post adaptive card action

13. Click **New step**.
14. Select the **Wait for an approval** Approvals action.
15. In the **Approval ID** field, select the **Approval ID** dynamic content token under **Create an approval**:

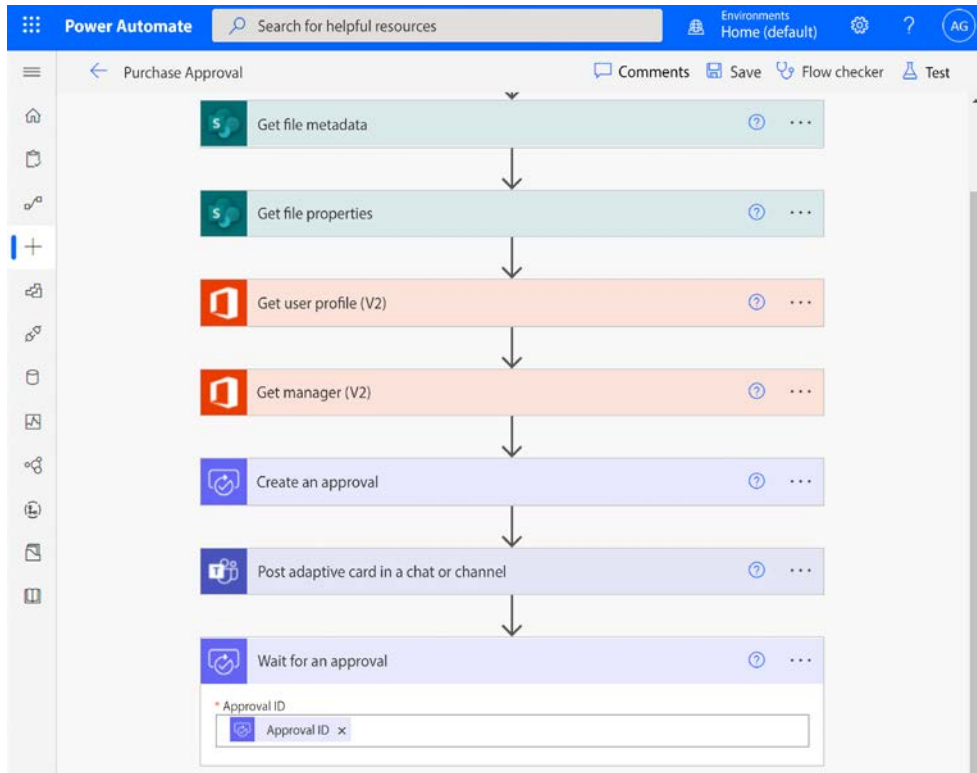


Figure 11.13: Configuring the Wait for an approval action

16. Click **Save**.

You've now configured an approval workflow to begin, post an adaptive card to the Teams user, and wait for their response. In the next section, we'll evaluate the response and decide what actions to take.

## Returning the response

In this final section, we'll configure the actions (notifying the requester and moving the submitted file) based on the approver's response. Since there are a lot of steps, we'll divide it into smaller chunks.

First, we'll configure what happens if the approver decides to approve the request.

To complete this part of the flow, ensure that the flow from the previous section is open for editing, and follow these steps:

1. After the **Wait for an approval** action, click **New step**.
2. Select the **Condition** control.
3. In the **Choose a value** box on the left side of the control, select the **Outcome** dynamic content token:

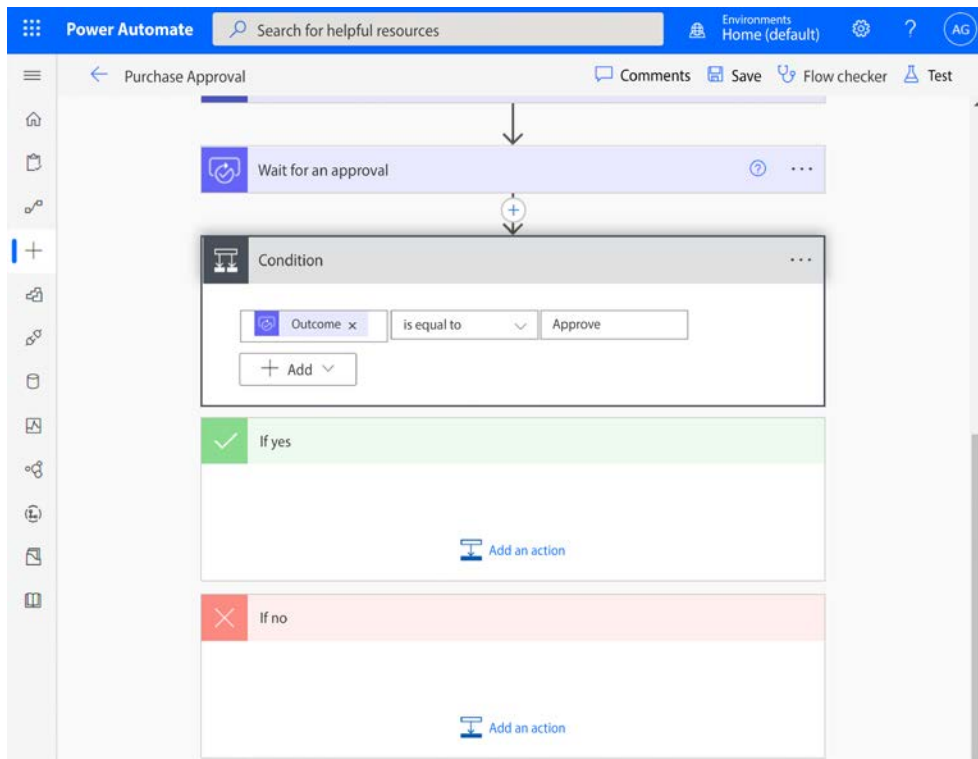


Figure 11.14: Configuring the condition control

4. Select the **is equal to** evaluation method for the condition.
5. Enter the text Approve in the right-hand box of the condition.
6. In the **If yes** branch of the condition, click **Add an action**.
7. Select the **Post a message in a chat or channel** action:

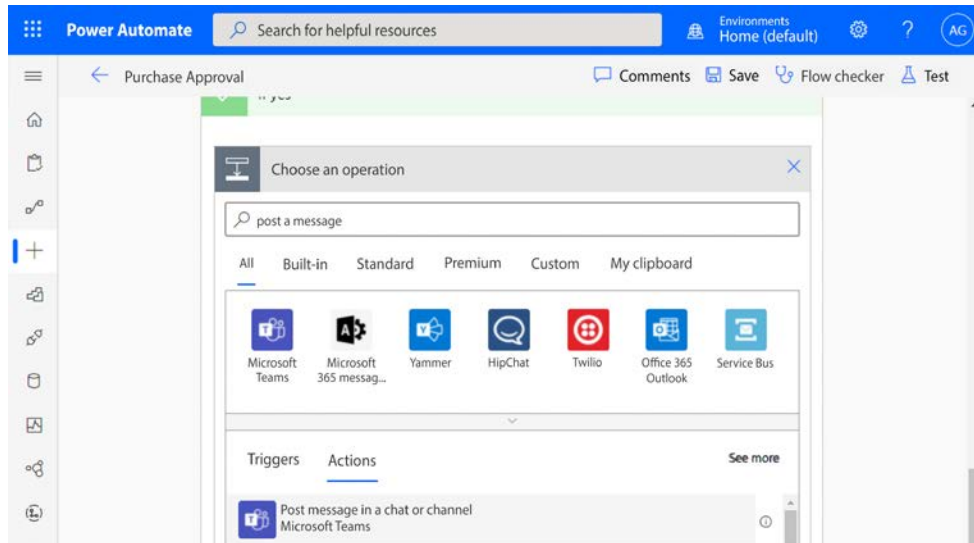


Figure 11.15: Adding the Post message action

8. In the **Post as** section, choose **Flow bot**.
9. In the **Post in** section, choose **Chat with the Flow bot**.
10. In the **Recipient** field, add the **Mail** dynamic content token under the **Get user profile** section. This will send the response to the *requester*.
11. In the **Message** field, enter a value that will be posted in the body of the message to the *requester*.
12. In the **If yes** branch, select **Add an action**.

13. Select the **Move file** SharePoint action:

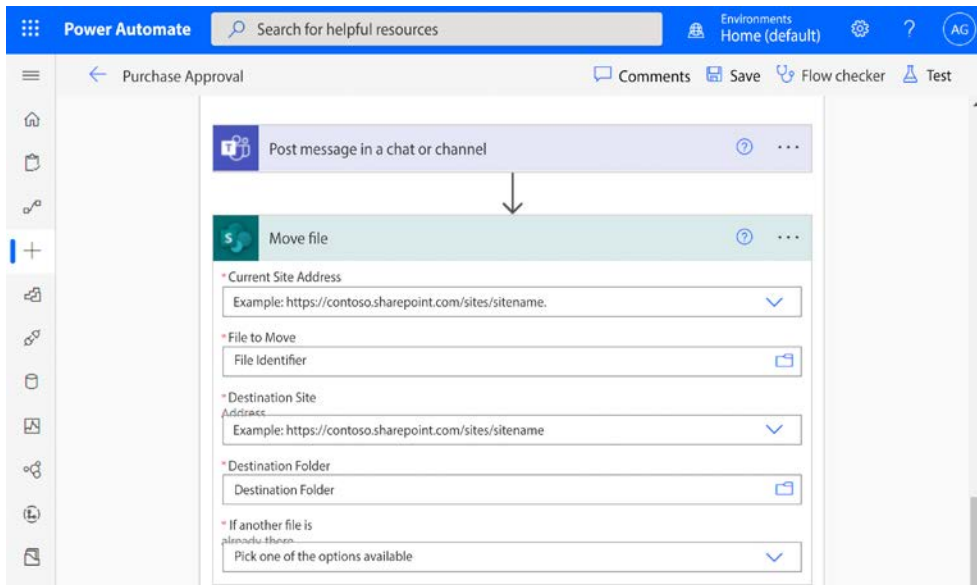


Figure 11.16: Adding the Move file SharePoint action

14. In the **Current Site Address** field, select the SharePoint site used in the flow.
15. In the **File to Move** field, select the dynamic content token **x-ms-file-id** from the **When a file is created in a folder** section.
16. In the **Destination Site Address** field, select the SharePoint site used in the flow.
17. For **Destination Folder**, select the **Approved** folder in the default document library.

18. Finally, select an action to take if an existing file with the same name exists in the destination folder. Select an option under the **If another file is already there** dropdown, such as the **Move with a new name** action. The completed branch should look similar to *Figure 11.17*:

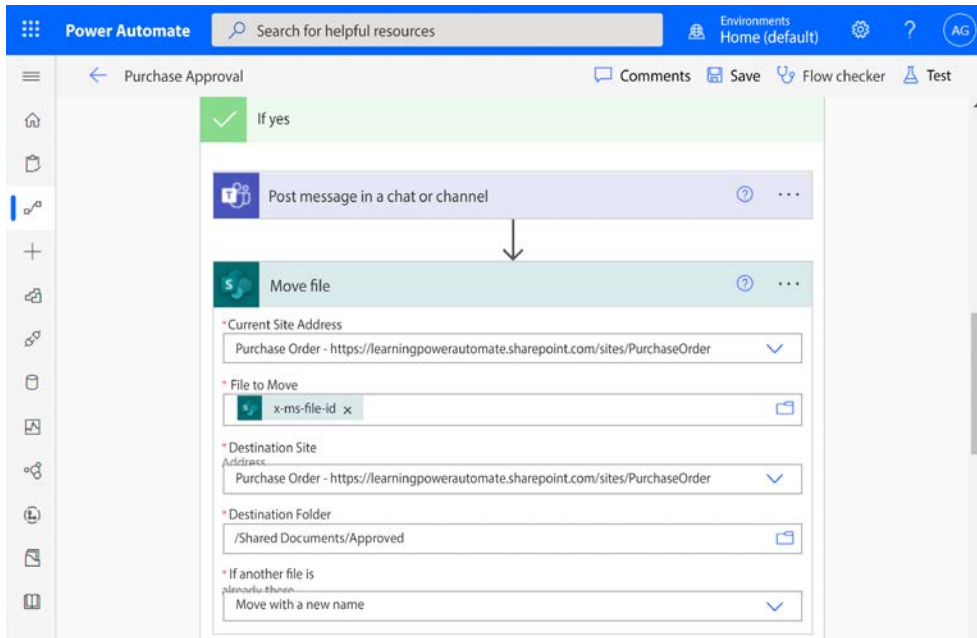


Figure 11.17: Configuring the SharePoint Move file action

19. Click **Save**.

Now, we'll move on to configuring what happens when the approver rejects or denies the request. The steps are quite similar—in fact, it's the same set of actions, just editing the message responses and file destinations. You can repeat the steps and update them, or you can save yourself some time with the **Copy to my clipboard** feature:

1. In the **If yes** branch, select the ellipsis for the **Post message in a chat or channel** action and then select **Copy to my clipboard (Preview)**:

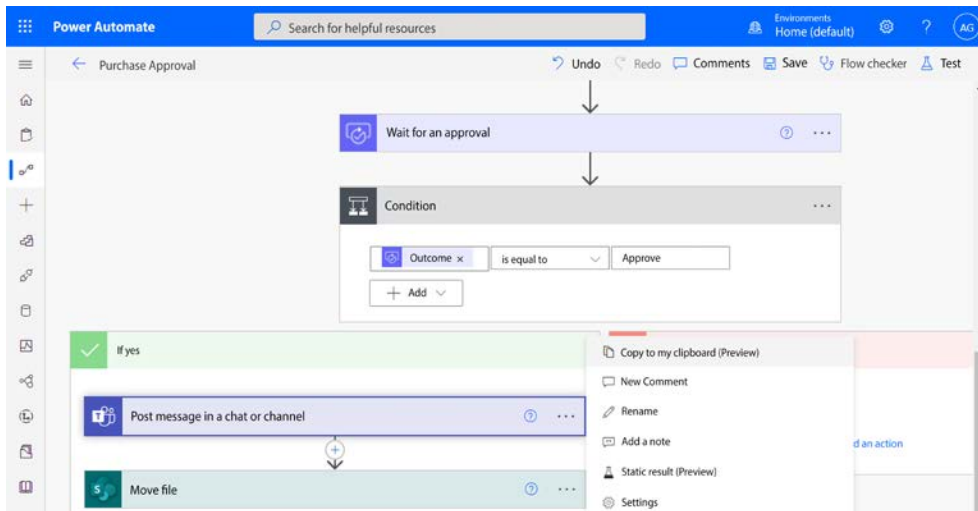


Figure 11.18: Copying an action

2. In the **If no** condition branch, select **Add an action**.
3. In the **Choose an operation** dialog, select the **My clipboard** tab, and then select the **Post message in a chat or channel** action:

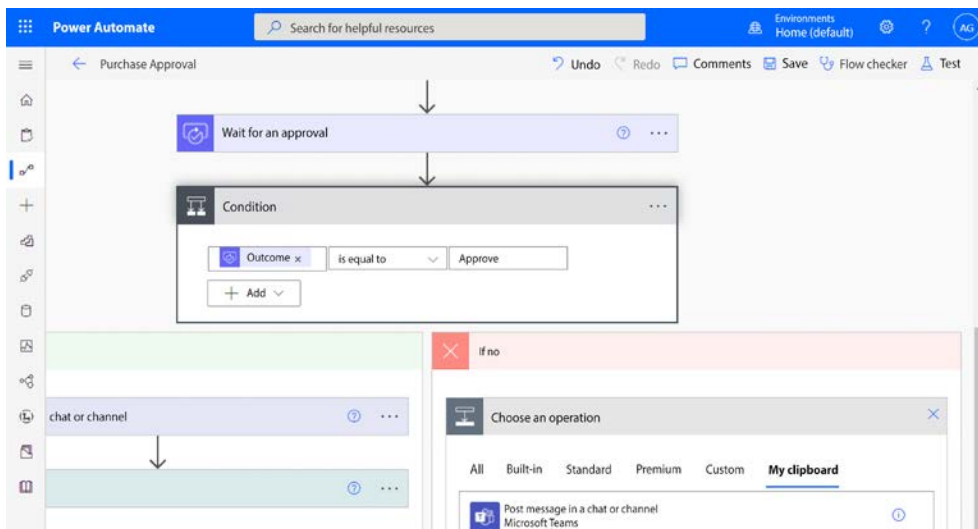
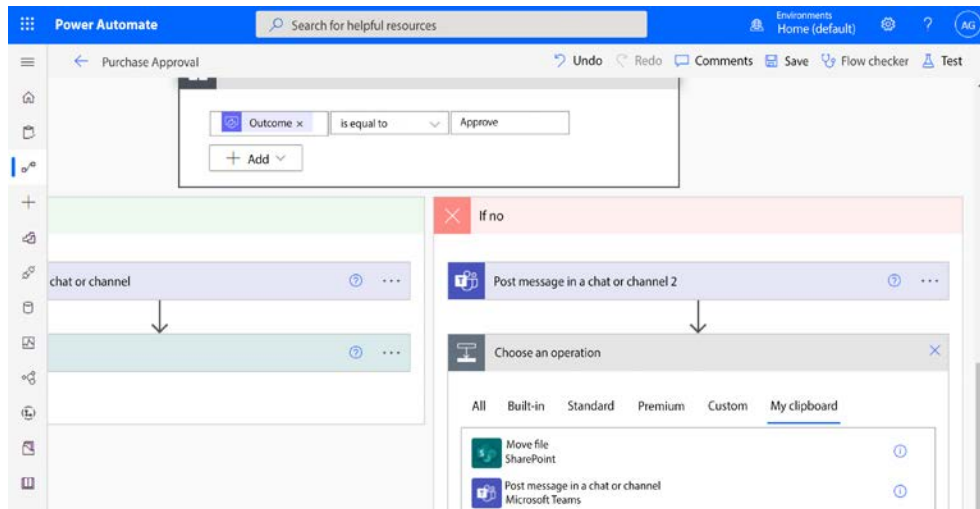


Figure 11.19: Selecting an action to insert

4. Click the actions title bar to expand it. Notice how the fields are populated with the content of the copied action.
5. In the **Message** field, update the existing message value that will be posted in the body of the message to the *requester*. As it was when you configured the original action, you can use a combination of text, expressions, and dynamic content.
6. In the **If yes** condition branch, select the ellipsis for the **Move file** action and then select **Copy to my clipboard (Preview)**.
7. In the **If no** condition branch, select **Add an action**.
8. In the **Choose an operation** dialog, select the **My clipboard** tab, and then select the **Move file** action, as shown in *Figure 11.20*:



*Figure 11.20: Selecting the Move file action to insert*

9. Select the **Move file** SharePoint action title bar to expand it.
10. For **Destination Folder**, select the **Rejected** folder in the default document library.



11. Finally, select an action to take if an existing file with the same name exists in the destination folder. Select an option under the **If another file is already there** dropdown, such as the **Move with a new name** action. The completed branch should look similar to *Figure 11.21*:

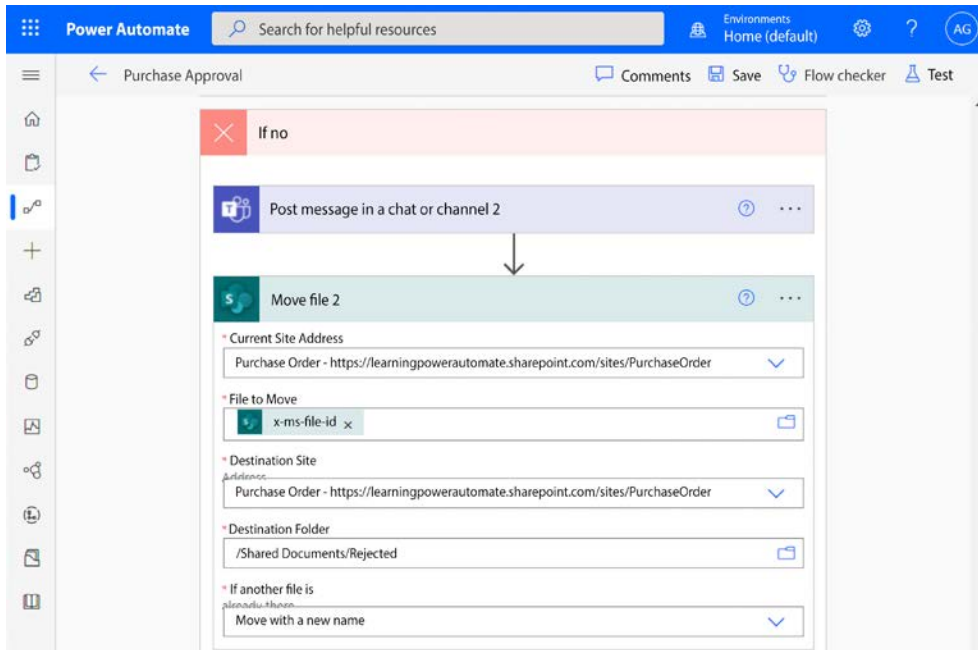


Figure 11.21: Reviewing the configuration of the If no branch

12. Click **Save**.

You've now configured the flow to post a message to the requester and move the file that triggered to flow to the **Approved** or **Rejected** folder, depending on the outcome.

Next, we'll test the flow!

## Testing the flow

In order to test this flow, you'll need to meet the following prerequisites:

- At least two users with Exchange Online and Microsoft Teams enabled
- Users should have the Power Automate app (formerly called the Flow bot app) enabled

- One of the users should be configured as a *manager* of the other users
- At least two browser sessions, one logged in as *approver* and one logged in as *requester*

You can use the Microsoft 365 admin center (<https://admin.microsoft.com>) to configure a manager relationship. Navigate to **Users | Active users**, select a user to edit, and click the **Edit manager** link to configure a manager:

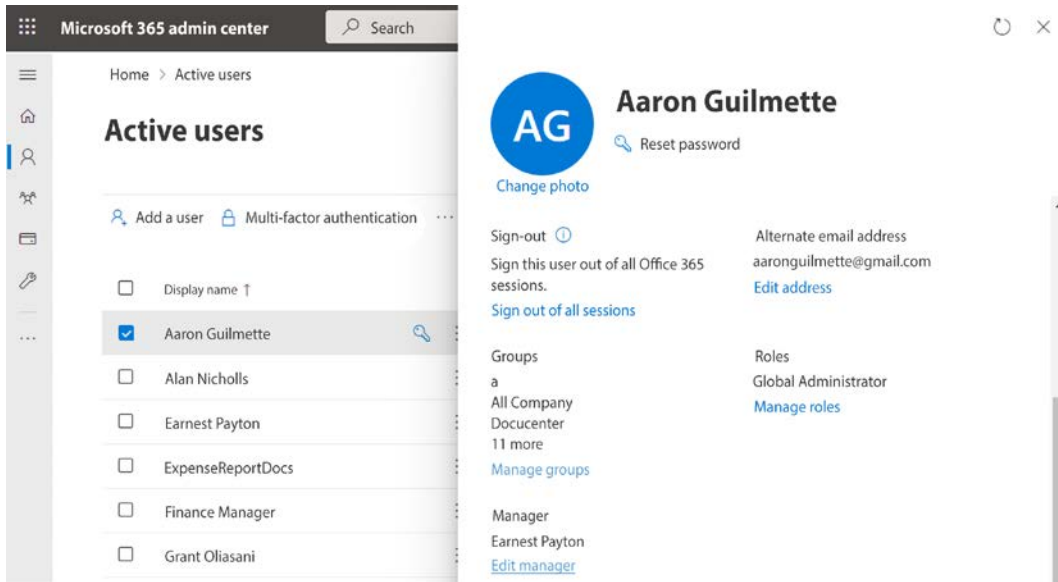


Figure 11.22: Configuring a manager



If you don't have the capability to set the manager relationship, you will need to edit the flow to use a particular email address of a user to use as the approver instead of the Mail token from the **Get manager (V2)** action.

The testing process will involve three core steps:

- Requesting approval
- Approving the request
- Reviewing the response

Let's start testing!

## Requesting approval

The process must be started by the requester. The requester must have the manager field populated in Azure AD. Once this requirement is met, follow these steps:

1. Using a separate browser, log in to Microsoft 365 as the *requester* user
2. Browse to the SharePoint site that is being monitored by the flow
3. Upload a file to the **Submitted** folder in the SharePoint site being monitored for a file

This activity will trigger the flow.

## Approving the request

For this process, you must have a separate browser session logged in as the *approver*:

1. When logged in to a browser session to Office 365 as the *approver*, open Microsoft Teams (<https://teams.microsoft.com>).
2. Select **Chat** from the left rail and select the message from the Power Automate app.
3. To approve or reject the request, click the **Approve** or **Reject** button on the adaptive card:

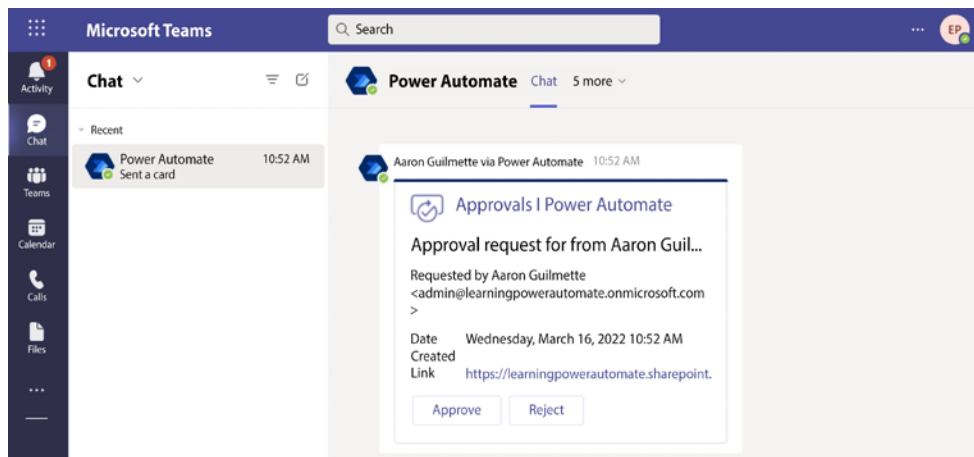


Figure 11.23: Viewing the adaptive card

4. Optionally, enter approval or rejection comments. Click **Submit**.

Once this is done, it will trigger the next part of the flow.

## Reviewing the response

In this final step, the requester should be able to view the approver's response:

1. When logged in to a browser session to Office 365 as the *requester*, launch Teams (<https://teams.microsoft.com>)
2. Select **Chat** from the left rail and select a message from the Power Automate app
3. Review the message:

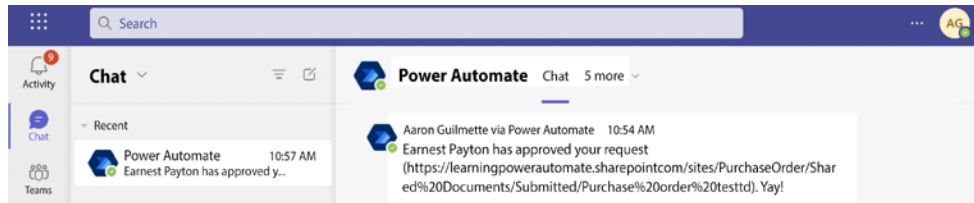


Figure 11.24: Reviewing the approval message from Teams

The approval process has been completed.

You can also use additional verification methods such as reviewing the flow's run history and browsing to the SharePoint document library to confirm that the file has been moved:

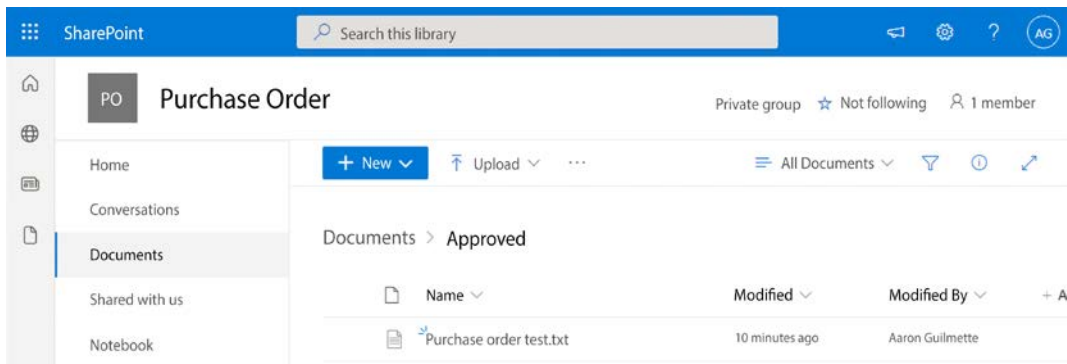


Figure 11.25: Confirming that the file has been moved to the Approved folder

The adaptive cards and flow messages can be customized. As you gain familiarity and confidence with dynamic content value tokens, formatting, and expressions, you can build on these examples to make cards and messages more personalized and detailed.

## Summary

In this chapter, you built on the knowledge that you've gained throughout the book to build an approvals integration into Microsoft Teams. You used familiar components, such as the SharePoint file creation trigger and conditions, and expanded into using adaptive cards and the Flow bot for Microsoft Teams. Using this type of design keeps approvers from task-switching or application-switching out of Microsoft Teams to complete approvals, which will help your organization improve productivity.

Finally, you were able to test the flow and see it from the perspective of both the requester and the approver.

In the next chapter, you'll learn how to start incorporating databases into flows. This will open a lot of opportunities to interact with line-of-business applications and create complex, business-driven automations.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 12

## Using a Database

Databases are critical infrastructure components that are used to compile information about tasks, individuals, services, products, or any other entity that needs to be tracked. Common database-driven applications include **Customer Relationship Management (CRM)** platforms, product catalogs, **Enterprise Resource Planning (ERP)** applications, accounting packages, storefronts, and inventory management tools.

In the world of Power Automate, you might use a database to store information about responses to surveys or forms, or to manage scheduled jobs. You may even use one as a logging repository for actions taken by other applications and users. In this chapter, we're going to look at some of the basics of connecting to a SQL database and using a flow to add data entries. Specifically, we'll learn about the following topics:

- Understanding database connectors, triggers, and actions
- Connecting to a database
- Creating a connection to a database
- Adding content to a database

By the end of this chapter, you will be able to understand the basics of how to connect and send data to a database with Power Automate.

Off we go!

## Understanding database connectors, triggers, and actions

If you're not familiar with databases, that's ok! While designing scalable database systems can be quite complex, the general concepts of databases are relatively simple.

Databases are typically made up of tables, columns, and rows. You can think of a simple database as a collection of spreadsheets, where the spreadsheet itself is the table, and then the rows and columns in the spreadsheet represent rows and columns in the database.

When reading or writing to a database, you need to tell the application some key details:

- The server name or address
- The database name
- Authentication credentials

This information is stored in a connection object or connection string and tells the application how to locate the database. Later, when performing actions such as queries, inserts, or updates against a database, you will specify additional details such as table names and filters to locate records or to tell the application where to send its data.

Power Automate can connect to a number of databases, such as Microsoft SQL Server, Azure SQL Data Warehouse, PostgreSQL, MySQL, Db2, Oracle, and Xooa blockchain, as well as Dataverse, which we discussed back in *Chapter 9, Getting Started with Approvals*. These database technologies shares similar mechanics and concepts, though some terminology or procedures may be different. In the examples provided in this book, we're going to focus on using Microsoft SQL Server databases and the related connector, which is the standard to use when working in M365 environments. Although this is the most common database that you will encounter, you may want or need to use others depending on what you have available.

As with other connectors, the SQL connector has both triggers and actions. We'll examine these next.

### Triggers

The Microsoft SQL Server connector supports two triggers (at the time of writing):

- **When an item is created (v2)**
- **When an item is modified (v2)**

When working with both of these triggers, you'll need to supply a server name, database name, and table name, as well as optional select and filter query statements to help refine your queries.

## Actions

When working with a database, there are many actions you can perform. The most common actions that you'll work with are detailed in the following list:

- **Get tables:** List tables in a database
- **Get row(s):** Retrieves one or more rows from a table
- **Insert row:** Adds a new row to a table
- **Update row:** Modifies an existing row in a database table
- **Delete row:** Deletes a row from a table
- **Execute a SQL query:** Runs a SQL query
- **Execute a stored procedure:** Runs an existing stored procedure

You'll use the built-in Power Automate actions for manipulating rows and tables most often. However, if you need to perform more complex transactions or functions, you can use the *Execute a SQL query* and *Execute a stored procedure* actions.

Now that you know what triggers and actions are available, let's get connected to a database and make a simple transaction.

## Connecting to a database

In this section, we'll look at creating a simple button flow that adds a row of data to a database table. To do this, you'll need a database and a table. In the following example, we'll create a SQL instance and a database that we'll use for our Power Automate configuration.

### Creating a server

While you can use any supported database technology with Power Automate, we're going to focus on SQL Server. If you already have an existing SQL server available, you can skip this step. Otherwise, you can provision the necessary components in Azure to start working immediately.

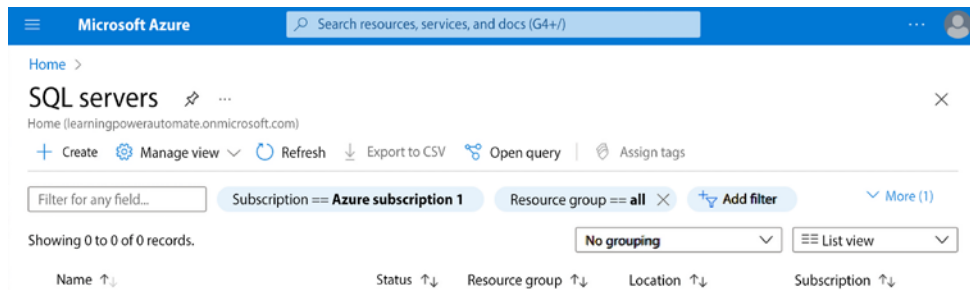


If you do not have access to server infrastructure, you can sign up for a free Azure trial at <https://azure.microsoft.com/en-us/offers/ms-azr-0044p/> or by navigating to <https://portal.azure.com>, signing in with your global admin credential for your trial Microsoft 365 tenant, and then completing the Azure trial sign-up process.



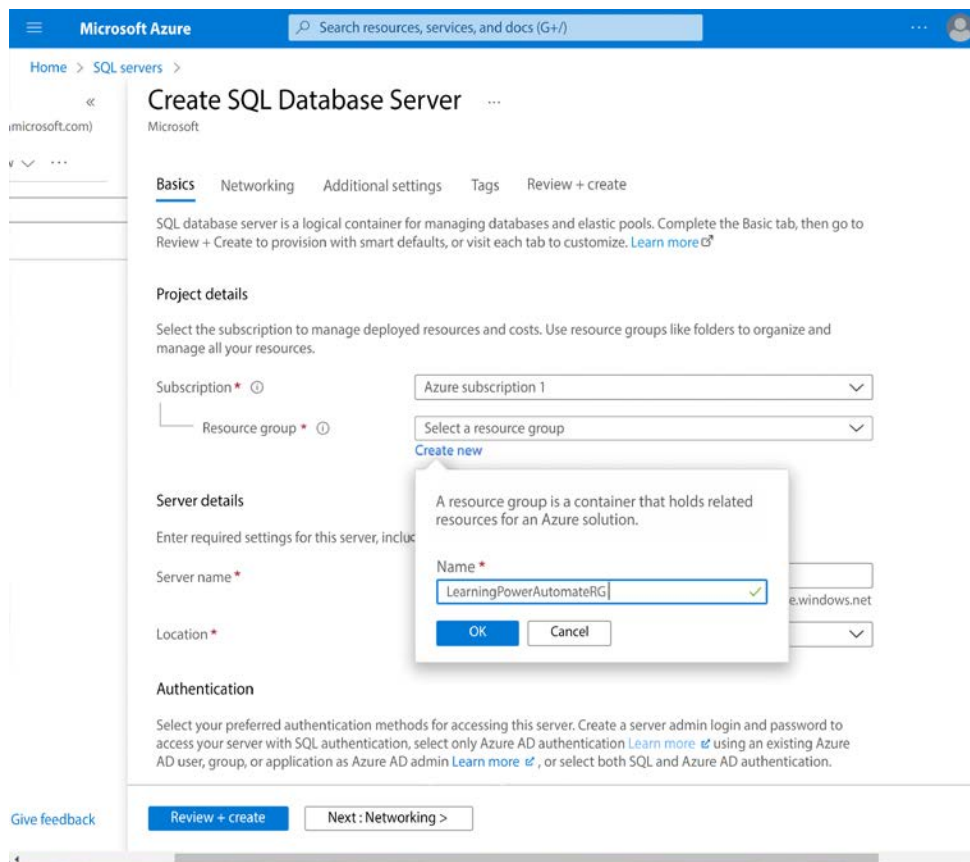
To create a SQL server in Azure, follow these steps:

1. Navigate to <https://portal.azure.com> and sign in using an account with administrative privileges.
2. From the search bar, begin typing SQL and select **SQL Servers**.
3. On the **SQL servers** blade, click **Create**:



*Figure 12.1: Creating a new SQL server*

4. Select a subscription and a resource group. A resource group is simply a logical container for components related to a project, service, application, or other administrative unit. If you do not have a resource group, you can create one, as shown in *Figure 12.2*:

*Figure 12.2: Creating a resource group*

5. Enter a unique **Server name**, select a data center **Location** near you, and credentials for administration. For purposes of the example, choose **Use SQL authentication**. When you're finished, click **Next : Networking**:

Microsoft Azure

Search resources, services, and docs (G+)

Home > SQL servers >

## Create SQL Database Server

Resource group \* (New) LearningPowerAutomateRG [Create new](#)

### Server details

Enter required settings for this server, including providing a name and location.

Server name \* learningpowerautomate ✓  
 .database.windows.net

Location \* (US) East US ✓

### Authentication

Select your preferred authentication methods for accessing this server. Create a server admin login and password to access your server with SQL authentication, select only Azure AD authentication [Learn more](#) using an existing Azure AD user, group, or application as Azure AD admin [Learn more](#), or select both SQL and Azure AD authentication.

Authentication method

- ☒ Use SQL authentication
- ☐ Use only Azure Active Directory (Azure AD) authentication
- ☐ Use both SQL and Azure AD authentication

Server admin login \* learning powerautomateadmin ✓

Password \* ..... ✓

Confirm password \* ..... ✓

[Give feedback](#) [Review + create](#) [Next : Networking >](#)

Figure 12.3: Configuring the database server properties

6. On the **Networking** tab, set the toggle to **Yes** to configure the firewall to allow access to Azure services.

7. Click **Review + create** to create your server.



Be sure to capture the name of the database server you create, as you'll need this later on when configuring the Power Automate connection object.

8. Review the final settings, and then click **Create**.

Wait while the Azure deployment finishes.

## Creating a database

Now that you have a SQL server instance provisioned, you'll need a database. If you already have a database that you can use, you can skip this step. Otherwise, you can use the following process to create a new database on the instance that you provisioned in the previous subsection:

1. From the Azure portal, click the search bar and begin typing **SQL databases**. Select the **SQL databases** option:

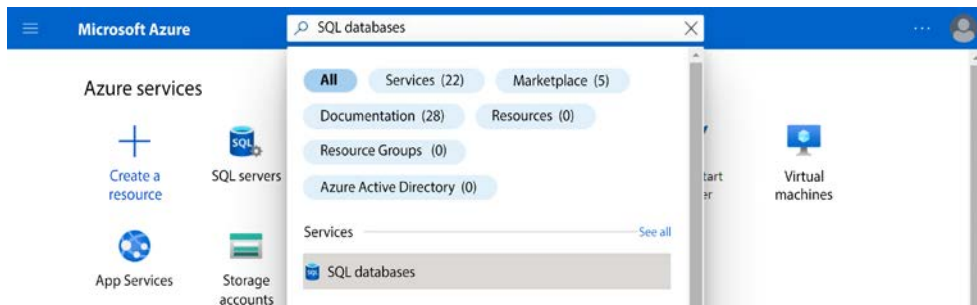


Figure 12.4: Searching for the SQL databases service

2. Click **Add** to create a new database.
3. Under **Project details**, select a subscription and a resource group. For simplicity's sake, use the same subscription and resource group that the server resides in.

4. Under **Database details**, enter a value for the new database name. Select a server. If you created a server in the previous section, select that server here. You can leave the rest of the settings configured to their defaults. Click **Review + create** to continue:

Microsoft Azure

Home > SQL databases >

## Create SQL Database

Microsoft

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \* ⓘ Azure subscription 1

Resource group \* ⓘ LearningPowerAutomateRG  
[Create new](#)

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name \* PowerAutomateDB ✓

Server \* ⓘ learningpowerautomate (East US)  
[Create new](#)

Want to use SQL elastic pool? \* ⓘ ☐ Yes ☒ No

Compute + storage \* ⓘ **General Purpose**  
Gen5, 2 vCores, 32 GB storage, zone redundant disabled  
[Configure database](#)

Backup storage redundancy

[Review + create](#) [Next: Networking >](#)

Figure 12.5: Configuring a new database



Be sure to capture the name of the database, as you'll need that when you are configuring the Power Automate connection object.

5. Click **Create**.

The wizard will complete the process and create a database.

## Creating a database table

In order for you to have a place to add content, the database will need a database table. You can create a database table by using the Query editor and following these steps:

1. From the Azure portal, click the search bar and begin typing SQL databases. Select the **SQL databases** option.
2. Select the database that you created in the previous subsection to open its properties page.
3. Select **Query editor (preview)**. Enter your login credentials to access the database and then click **OK**:

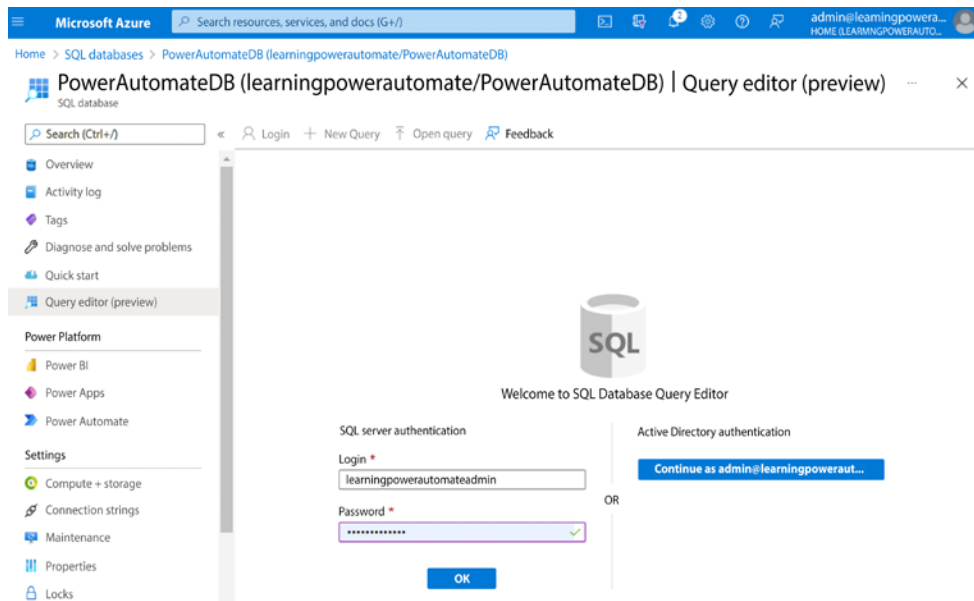


Figure 12.6: Logging into the database

- Note: If prompted, you may need to enable a firewall rule to allow your local computer to communicate with the server. Click the link in the error message, as shown in *Figure 12.7*, to automatically configure the firewall:

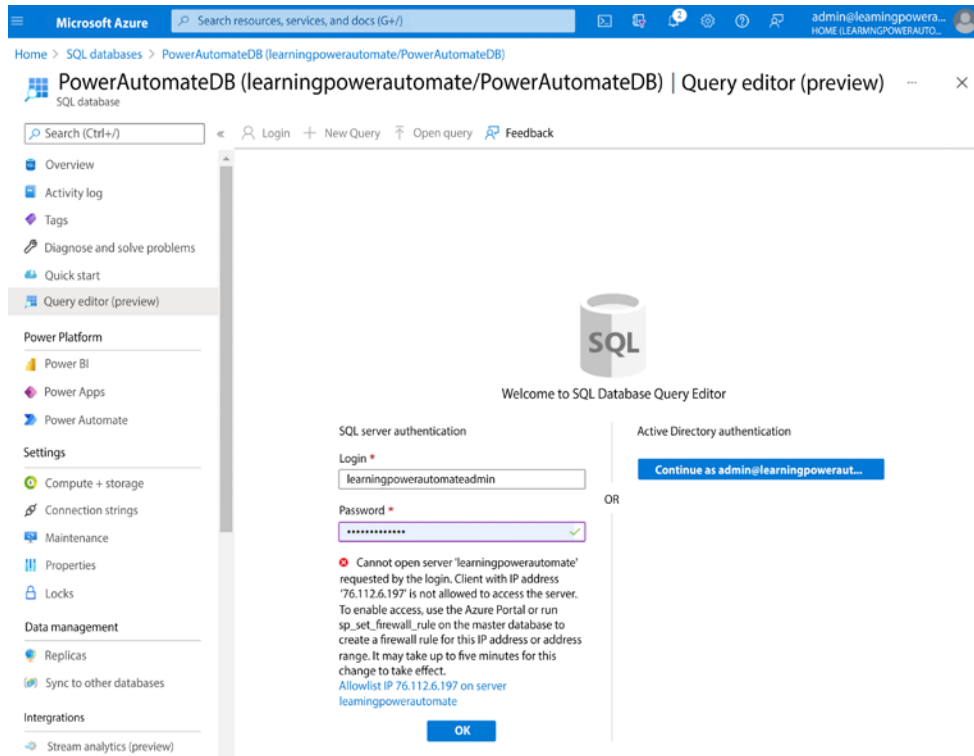


Figure 12.7: Prompt to create a firewall rule

- Next, we'll create a database table named **Customers** and insert a single row of data into it. Copy and paste the following code into the **Query editor**:

```
CREATE TABLE Customers
(
    CustomerID BIGINT IDENTITY,
    CompanyName VARCHAR(100),
    FirstName VARCHAR(100),
```

```

        LastName VARCHAR(100),
        JobTitle VARCHAR(50),
        Mail VARCHAR(75),
        TelephoneNumber VARCHAR(10)
    )
GO
INSERT INTO Customers VALUES('Contoso','John','Doe','Purchasing
Manager','jdoe@contoso.com','2485551234');
GO

```

6. Click **Run**:

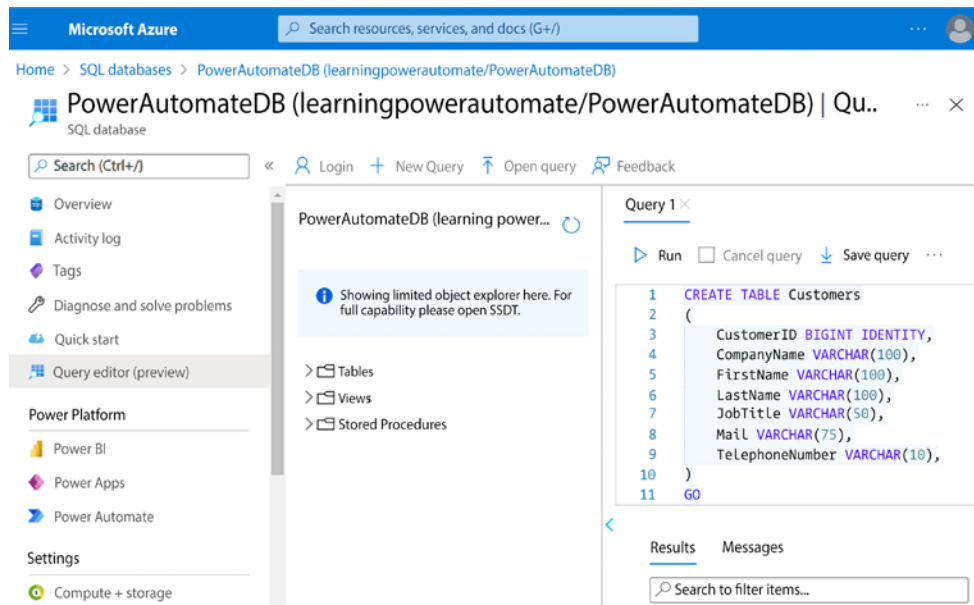


Figure 12.8: Ready to submit a query

7. Review the **Messages** pane to ensure that the database table has been created.

At this point, you should now have a server, a database, and a table that contains some content. This database table should have seven columns (CustomerID, CompanyName, FirstName, LastName, JobTitle, Mail, and TelephoneNumber) and one row of data.



To verify that the table contains data, expand **Tables**, select **dbo.Customers**, and click the ellipsis. Choose the **Select Top 1000 Rows** action, and then review the data in the **Results** pane, as shown in *Figure 12.9*:

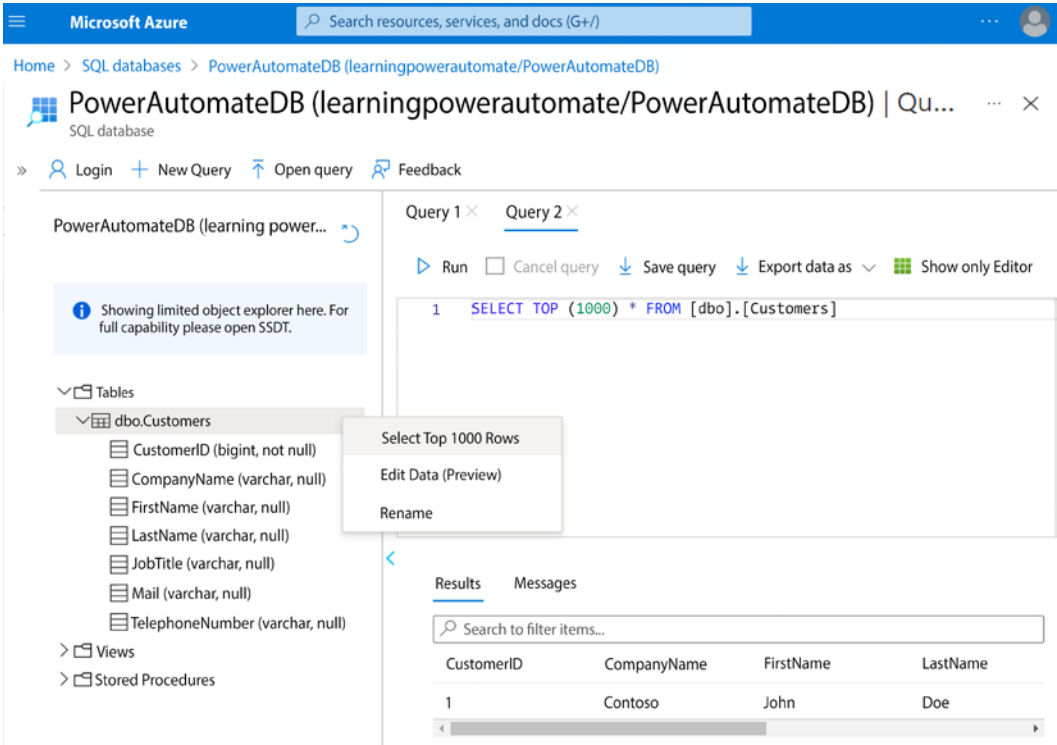


Figure 12.9: Reviewing table data

If everything was successful, you’ll have a sample database configured and ready to use as a destination for Power Automate.

Next, we’ll establish a connection to the database in Power Automate.

## Creating a connection to a database

As you learned earlier in the chapter, when working with databases, you’ll need to tell an application how to connect to a database. That data is stored in a configuration object generally called a connection string. In Power Automate, the connection string data is referred to as a **connection**.

Connection details will depend on the type of database you’re connecting to, but the most common fields or properties that you’ll need to populate include a server name or IP address, credentials, port numbers, and a database name.

To connect to a database in Power Automate, follow these steps:

- 1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>). Expand **Data** and click **Connections**.
- 2. Select **New connection**:

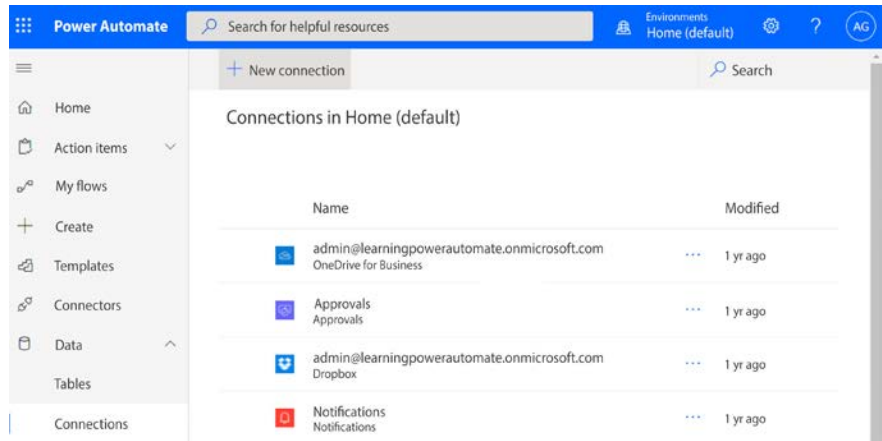


Figure 12.10: Creating a new connection

- 3. Select **SQL Server** from the list of connection types:

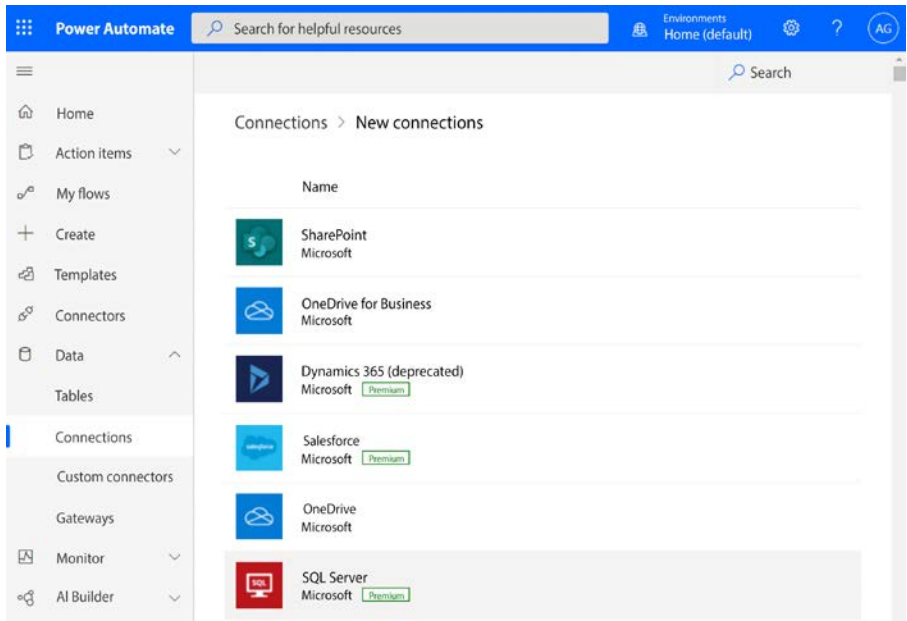


Figure 12.11: Selecting the SQL Server connection type

4. Fill out the dialog box using the details you saved earlier, including the authentication type, SQL server name, and SQL database name:

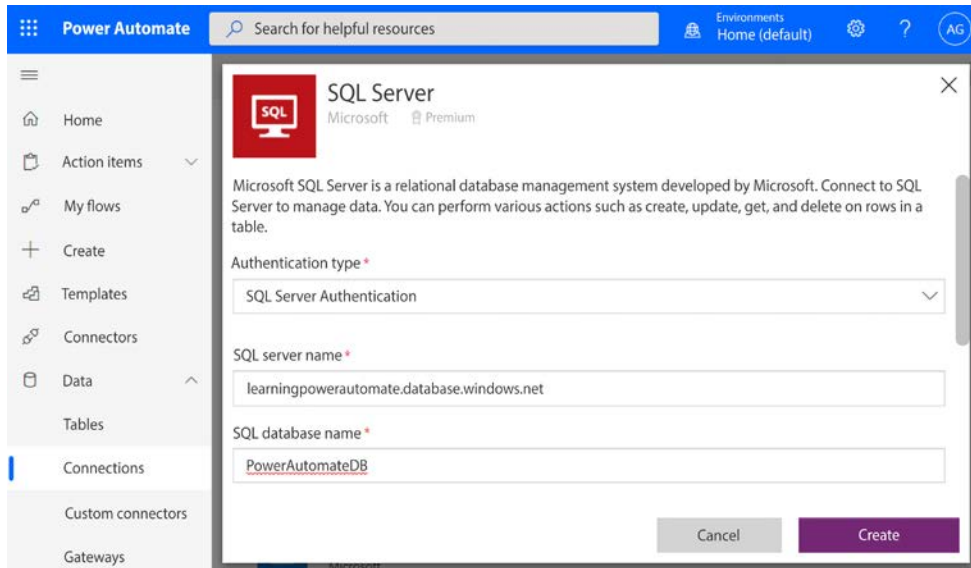


Figure 12.12: Configuring the database connection

5. Scrolling down the dialog box shown in *Figure 12.12*, enter the credentials saved from earlier and leave the **Choose a gateway** dialog selection box blank, because we haven't configured a data gateway.
6. Click **Create**.

Now that a connection has been established, you can create a simple button flow in order to place information in the database.

## Adding content to a database

Now that you have configured a server, a database, and a table with columns, you get to see all the pieces working together. In this section, you'll create a flow, execute it, and then verify that it wrote the data successfully.

First, let's create a flow so that we can add content.

## Creating the flow

To verify that you've configured database connectivity, we'll walk through creating a sample button or instant flow so that we can post data to the database. Follow these steps to create the flow:

1. From the Power Automate web portal (<https://flow.microsoft.com>), select **Create**.
2. In the **Start from blank** section, select **Instant cloud flow**.
3. Enter a name for the flow, and then select **Manually trigger a flow** as the trigger type.
4. Click **Create**.
5. Click **New step**.
6. In the search box, enter `SQL insert` and then select the **Insert row (V2)** action:

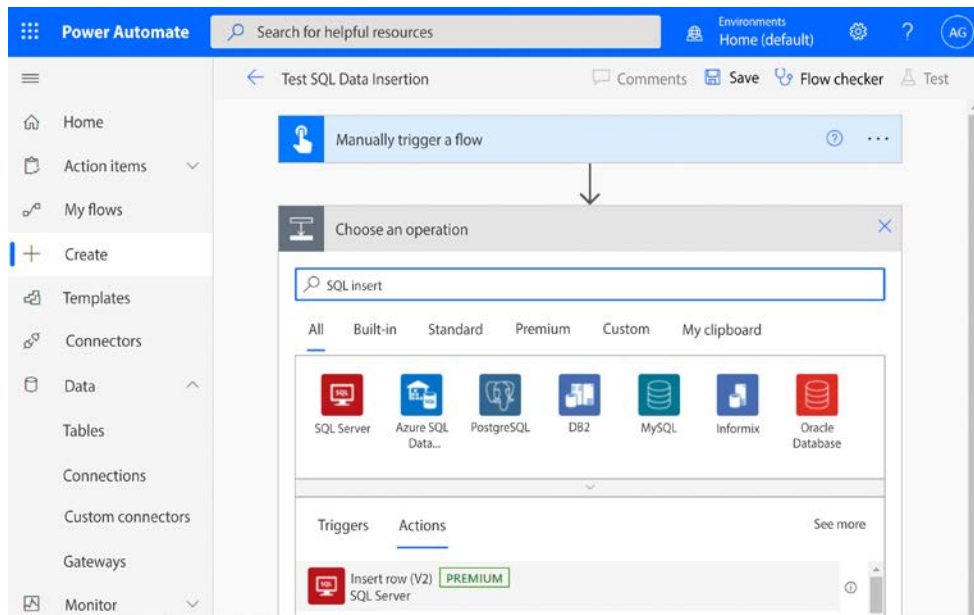


Figure 12.13: Selecting the Insert row (V2) action

7. From the dropdowns for the action, select the server name, database name, and table name you created in the *Creating a database table* section:

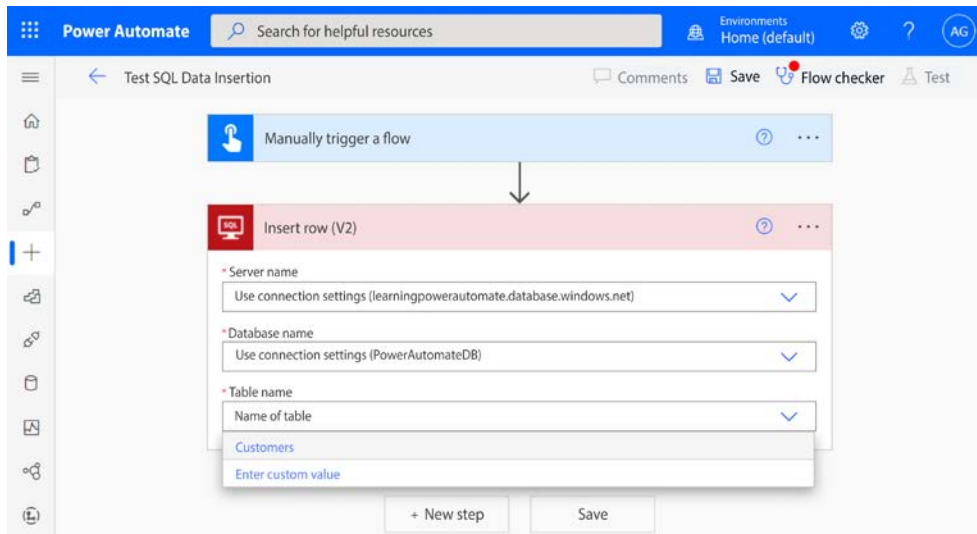


Figure 12.14: Selecting the server, database, and table

8. Once you've selected the items from your connection, the action will update to display all the available columns. Add some test data, as shown in Figure 12.15:

The screenshot shows the Microsoft Power Automate web interface. At the top, the header includes the 'Power Automate' logo, a search bar, and navigation links for 'Environments', 'Home (default)', and a user profile icon labeled 'AG'. The main workspace is titled 'Test SQL Data Insertion' and includes buttons for 'Comments', 'Save', 'Flow checker', and 'Test'. A blue bar at the top of the workspace indicates the flow is triggered 'Manually trigger a flow'. Below this, an 'Insert row (V2)' action is added. The action's configuration is as follows:

Property	Value
Server name	Use connection settings (learningpowerautomate.database.windows.net)
Database name	Use connection settings (PowerAutomateDB)
Table name	Customers
CompanyName	Fabrikam
FirstName	Dan
LastName	Jump
JobTitle	IT Manager
Mail	djump@fabrikam.com
TelephoneNumber	4255551212

Figure 12.15: Adding test data

9. Click **Save**.

The flow is now ready to test.

## Executing the flow

Use the process you learned about in *Chapter 5, Creating Button Flows*, to execute the flow. As a refresher, you can follow these steps:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>) and select **My flows**.
2. Hover over the **Test SQL data Insertion** flow and click the **Run** button to execute the flow:

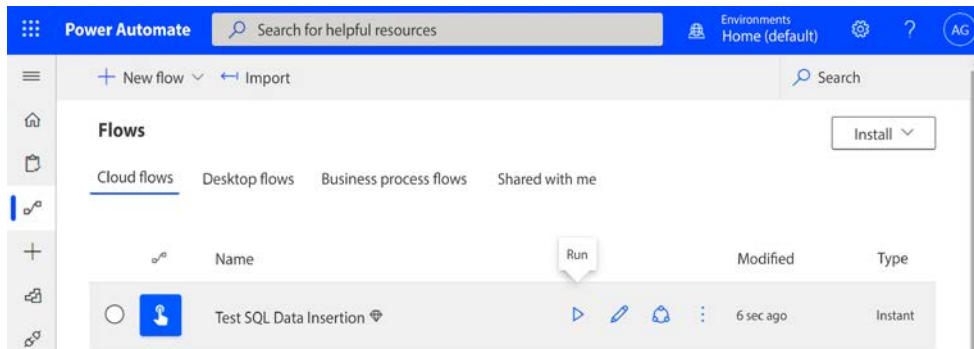


Figure 12.16: Running the flow

3. On the **Run flow** flyout panel, click **Continue**.
4. Click **Run flow**.
5. Click **Done**.

With that, the flow has been successfully executed.

## Verifying the flow

You can check whether the content was written to the flow by examining the flow's run history or by querying the database table in SQL.

## Reviewing the run history

The easiest and quickest way to verify whether the flow is successful is to examine its run history. The flow's run history will show the steps that were performed during the flow's execution. You can review the run history for the flow by using the following process:

1. Expand the ellipsis for the flow and then select **Run history**.
2. Select the date and time of the run to display the history.

3. Expand the SQL action to view the data:

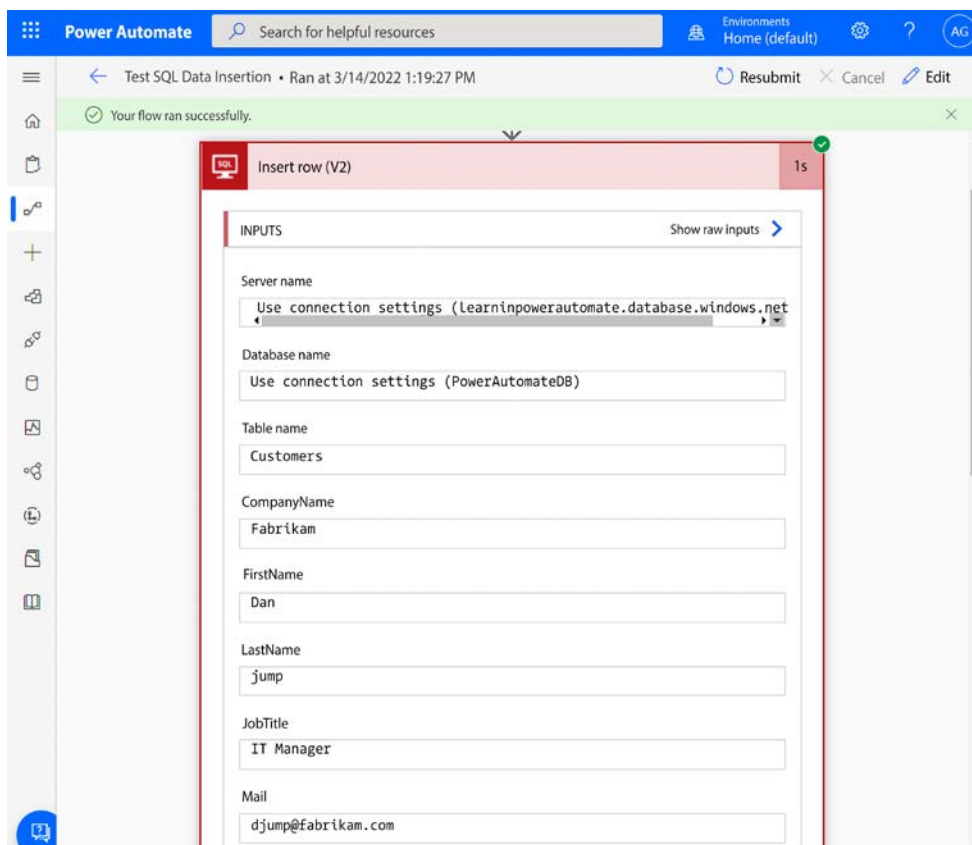


Figure 12.17: Viewing the SQL Insert row (V2) action data

The expanded action will show the data that was inserted into the table.

## Reviewing the SQL data

Using the SQL Query editor, you can also view the newly added data. To view the SQL data directly, follow these steps:

1. From the Azure portal (<https://portal.azure.com>), navigate to **SQL databases**.
2. Select **Query editor (preview)**. Enter your login credentials to access the database, if necessary, and then click **OK**.



- 3. In the **Query editor** blade, expand **Tables**, select **dbo.Customers**, and then click the ellipsis for the table. Click **Select Top 1000 Rows**:

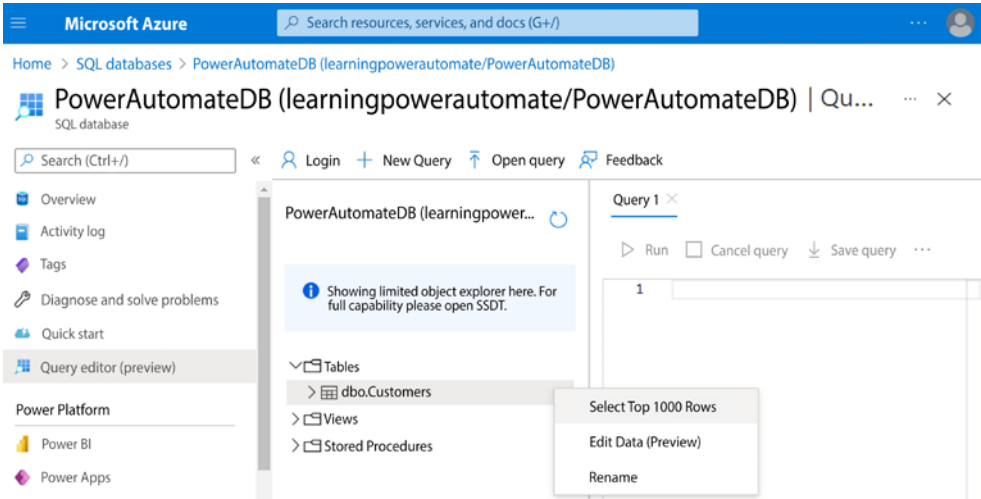


Figure 12.18: Navigating to the *dbo.Customers* database table

- 4. Compare the data in the **Results** pane to the values you entered in the *Creating the flow* section:

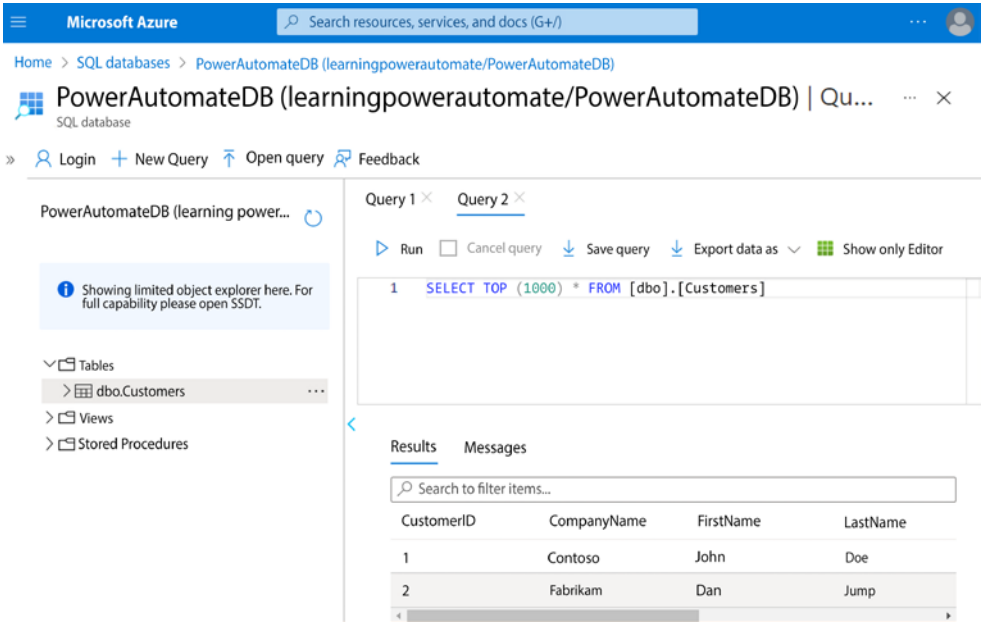


Figure 12.19: Reviewing data in the *Results* pane

No matter how you choose to verify that the flow was executed successfully, you should be able to see the same data.

## Summary

In this chapter, you learned about some of the database interaction capabilities of Power Automate. Following the steps in this chapter, you were able to successfully create an Azure SQL Server instance, a database, and a database table to store content. You were able to connect to the database, create a flow, and insert a new row of data. Finally, you examined the run history of the flow and learned how to query the database directly.

The next chapter will build on this knowledge by showing you how to take Microsoft Forms data and insert it into a database using Power Automate.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>





# 13

## Working with Microsoft Forms

Gathering and processing feedback from users or customers is an important part of business operations. Microsoft Forms is a survey tool that can be used to capture that information, both with fixed questions and free-form text entry options.

In this chapter, we're going to build on the concepts we've already learned regarding conditions (*Chapter 8, Working with Conditions*) and adding content to a database (*Chapter 12, Using a Database*) to create flows based on input from Microsoft Forms and save it to a SQL database. Specifically, we'll look at the following topics:

- Understanding the Forms connector triggers and actions
- Creating a basic form
- Processing a form with Power Automate

When you finish this chapter, you'll have an understanding of how you can connect Forms and SQL with Power Automate.

Let's dig in!

### Understanding the Forms connector triggers and actions

Before we can begin crafting a flow that involves Microsoft Forms, it's important to understand what kinds of triggers and actions are available for the Forms connector. As a reminder, triggers are activities that can initiate a flow, and actions are the activities that a flow can perform. If you need to brush up on your terminology, you can refer back to *Chapter 1, Introducing Microsoft Power Automate*.

## Triggers

As mentioned in the introduction, Microsoft Forms is a survey and information gathering tool. An end user's interaction with Forms finishes when they complete and submit the form. As such, the Forms connector currently has only one trigger: **When a new response is submitted**. This trigger is activated when a user submits a form to the service.

## Actions

The Forms connector also has only a single action: **Get response details**. Using this action, Power Automate can retrieve the data submitted by the user and use the field values in a flow.

Before we can use Power Automate to process a form, we'll need a form to work with. In the next section, we'll create a form that will collect basic contact information.

## Creating a basic form

The Forms application is relatively straightforward to use. In this section, we'll create a basic form to collect user information. To create the form, follow these steps:

1. Launch the Forms application by navigating to <https://forms.office.com> and signing in.
2. If this is your first time signing into Forms, click the **Create a new form** button on the splash page. If you've logged into Forms previously, you can click the **New Form** button on the dashboard:

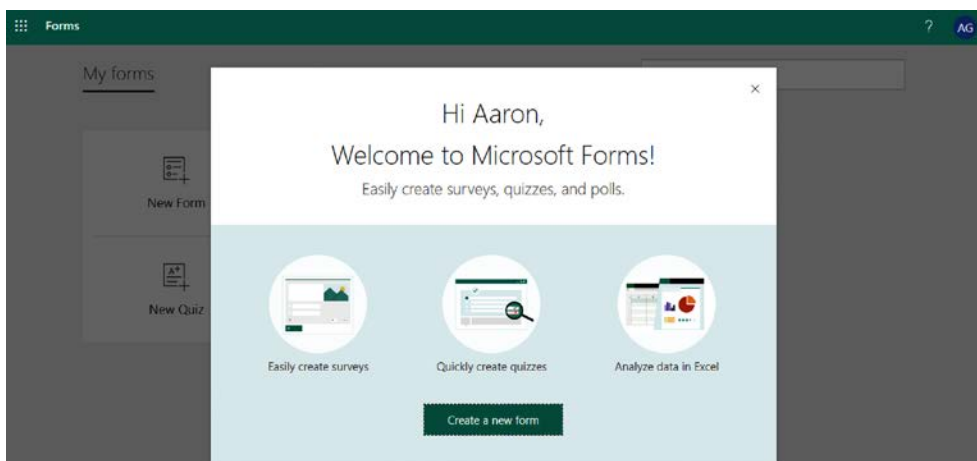
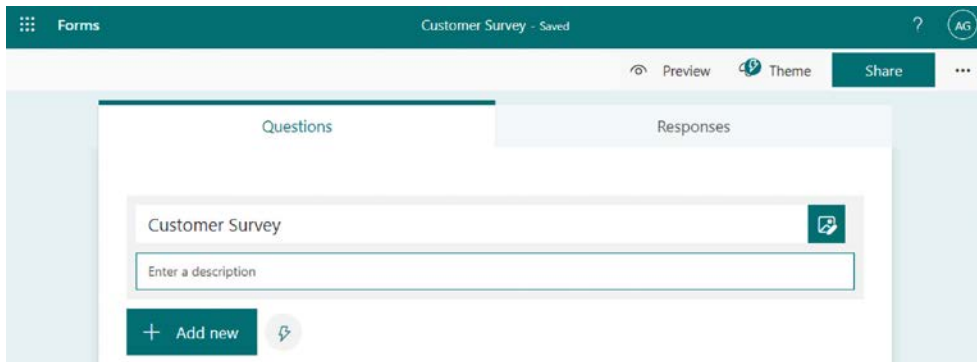


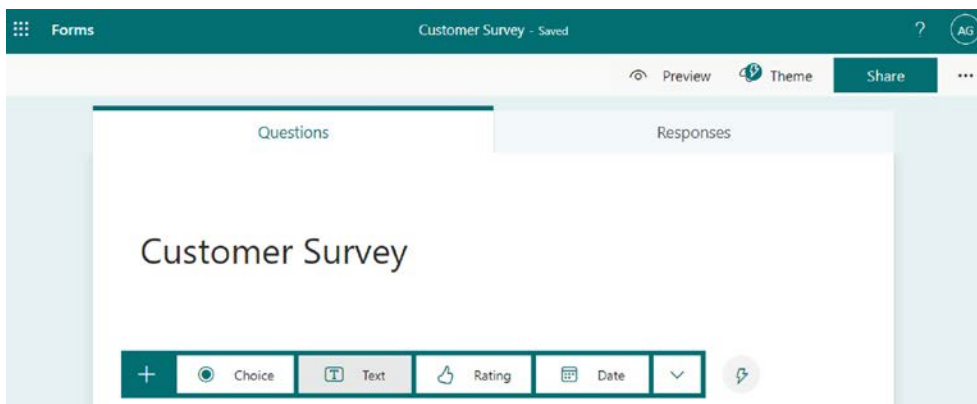
Figure 13.1: Microsoft Forms splash page

3. Click the title area (**Untitled form**) text box and enter a name for the form, such as **Customer Survey**, as shown in *Figure 13.2*:

The screenshot shows the Microsoft Forms editor interface. At the top, there's a teal header bar with 'Forms' on the left, 'Customer Survey - Saved' in the center, and a user profile icon 'AG' on the right. Below the header, there are tabs for 'Questions' and 'Responses'. The 'Questions' tab is active. In the center, there's a text box containing 'Customer Survey' and a smaller text box below it with the placeholder 'Enter a description'. At the bottom left of the question area, there's a '+ Add new' button and a share icon.

*Figure 13.2: Adding a title to the form*

4. Click **Add new** to add a new item.
5. Select the **Text** option:

The screenshot shows the Microsoft Forms editor interface after adding a new question. The 'Questions' tab is still active. The title 'Customer Survey' is at the top. Below it, there's a horizontal bar with a '+' icon on the left and a dropdown menu on the right. The dropdown menu is open, showing options: 'Choice' (selected with a radio button), 'Text' (with a 'T' icon), 'Rating' (with a thumbs up icon), and 'Date' (with a calendar icon). A share icon is also visible to the right of the dropdown.

*Figure 13.3: Selecting the text data type*

6. Enter the value `CompanyName`. You can enter something more descriptive if desired, but for the purposes of this example, make sure it is easily identifiable as the company name value. You can toggle the data field property to **Required**, if desired, as well as enabling field restrictions:

The screenshot shows the Microsoft Forms editor interface. At the top, there's a teal header bar with the word 'Forms' on the left, 'Customer Survey - Saving...' in the center, and a user profile icon 'AG' on the right. Below the header, there are three buttons: 'Preview', 'Theme', and 'Share'. The main content area has two tabs: 'Questions' (active) and 'Responses'. Under the 'Questions' tab, the title 'Customer Survey' is displayed. Below the title, there's a question card for '1. CompanyName'. The card has a text input field with the placeholder 'Enter your answer'. To the right of the input field is a small icon of a document with a pencil. Below the input field, there are two toggle switches: 'Long answer' (disabled) and 'Required' (enabled). To the right of the 'Required' toggle is a three-dot menu icon. At the bottom left of the question card, there's a teal button with a plus sign and the text 'Add new'.

Figure 13.4: Configuring the `CompanyName` data field

7. Repeat steps 4-6, adding fields for `FirstName`, `LastName`, `JobTitle`, `Mail`, and `TelephoneNumber`:

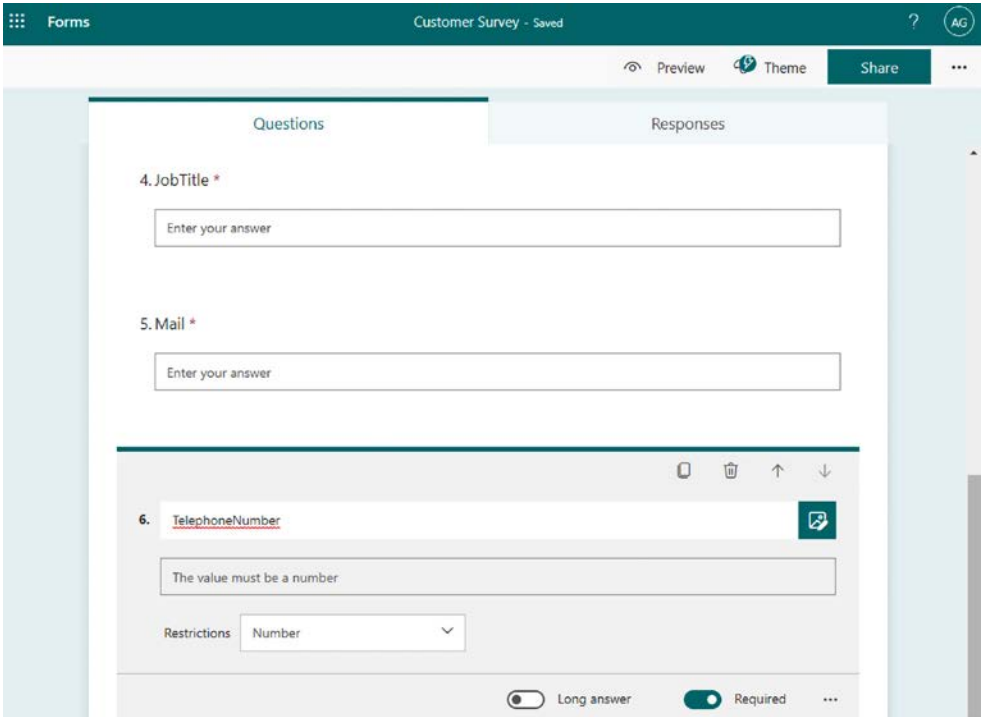


Figure 13.5: Configuring the remainder of the data fields

8. Click **Share**, and then in the URL box, select, copy, and save the value after the text **id=**. The value should look like a long random text string, such as:

```
XXkEUVeQ0SGJH73BSTmSE1XafnDwzBNuIS7TdkTq0dUME5KRjNTM1BMWDY4QUdMWVdI
U1RHSUE1MS4u
```

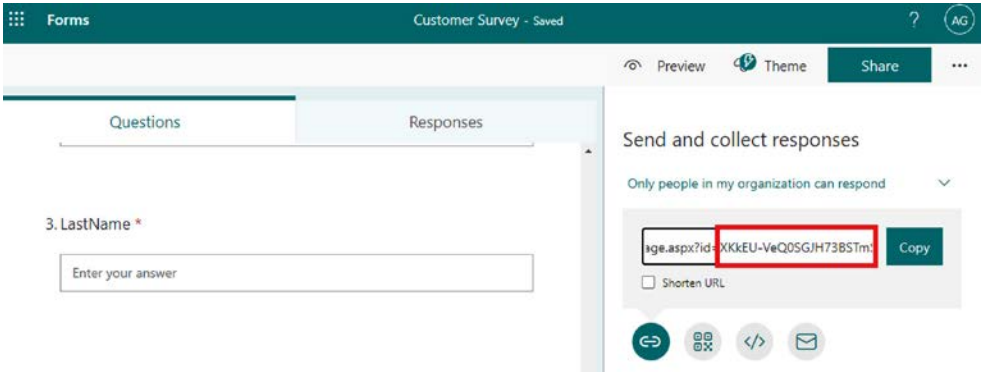


Figure 13.6: Copying the form ID



When you're finished, you should have a form with fields that resemble the column names for the database you created in *Chapter 12, Using a Database*, as well as the copied form ID value.

Next, we'll use Power Automate to save these form responses to a database for later analysis.

## Processing a form with Power Automate

In this exercise, we'll configure a flow to process the responses and save them into the database you created in *Chapter 12, Using a Database*. The flow will utilize the form ID value you saved in the previous exercise, the **When a new response is submitted** trigger, and the **Get response** details action for the Microsoft Forms connector.

Follow these steps to configure the flow:

1. Log into the Power Automate web portal (<https://flow.microsoft.com>) and click **Create**.
2. Under the **Start from blank** section, select **Automated cloud flow**.
3. Enter a name for the flow (such as Customer Survey) and select the **When a new response is submitted** trigger for Microsoft Forms. Click **Create**:

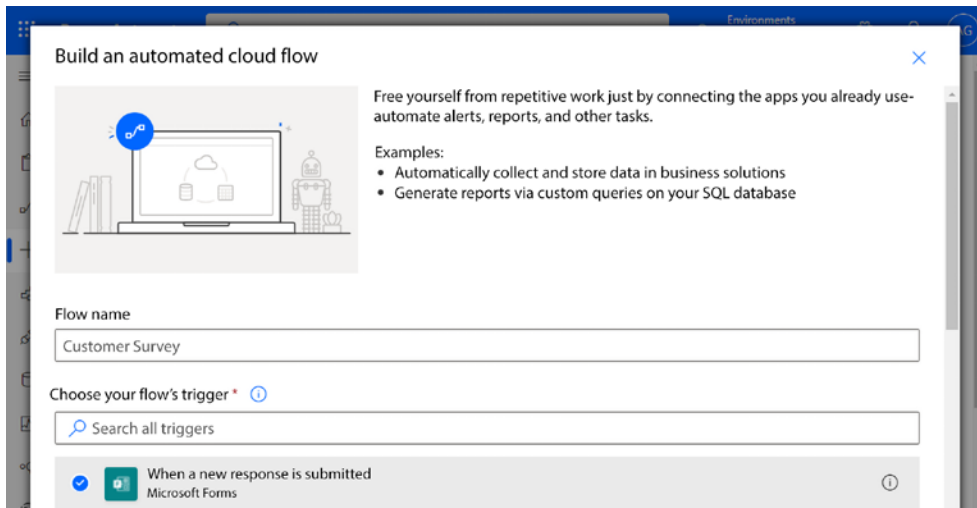


Figure 13.7: Creating a new automated cloud flow with the Forms trigger

4. In the details for the trigger, select the form if it is displayed. If the form you created is not displayed, select **Add a custom item** and paste in the form ID value you saved in the previous exercise:

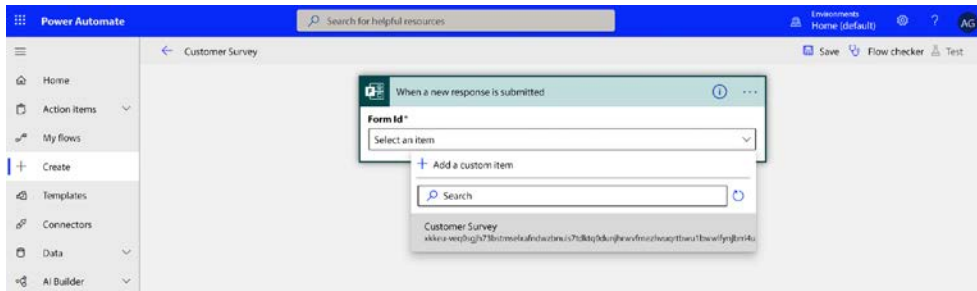


Figure 13.8: Manually entering the form ID value

5. Click **New step**.
6. In the search bar, start typing **Get response details** and select the action for Microsoft Forms:

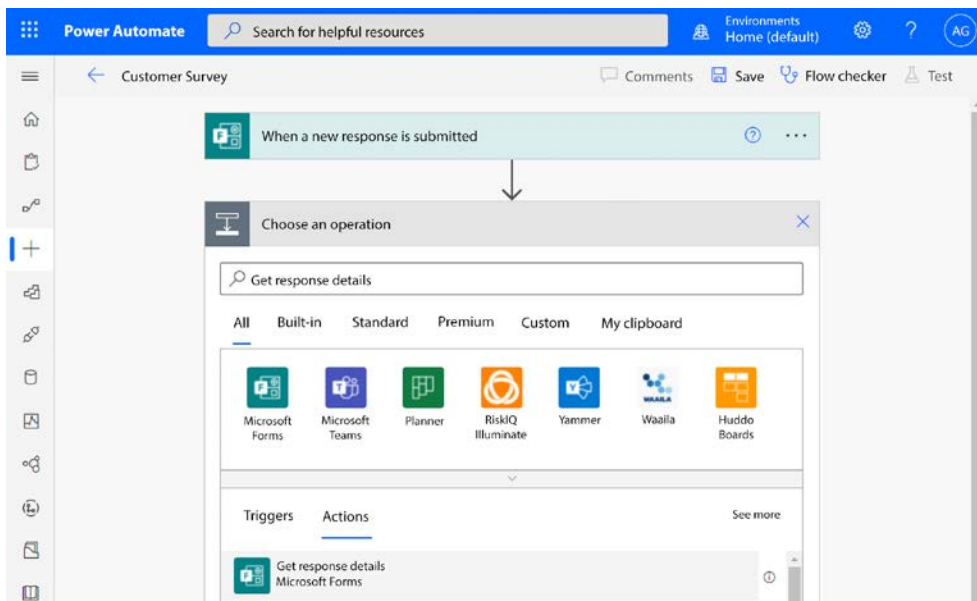


Figure 13.9: Adding the Get response details action

7. In the **Get response details** action, select the form ID representing the form (or the form name, if it is displayed). Select the **Response Id** box and then select the **Response Id** dynamic content token:

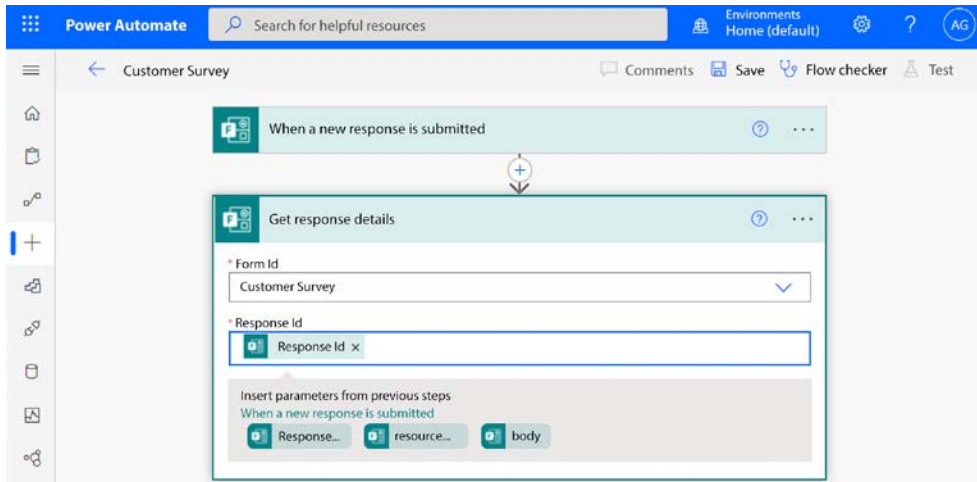


Figure 13.10: Adding the Response Id dynamic content value

8. Click **New step**.
9. Select the **Insert row (V2)** action for **SQL Server**.
10. Select the **Server name**, **Database name**, and **Table name** values from the connection settings you created in *Chapter 12, Using a Database*. After a moment, the column values should be displayed, as shown in the following screenshot:

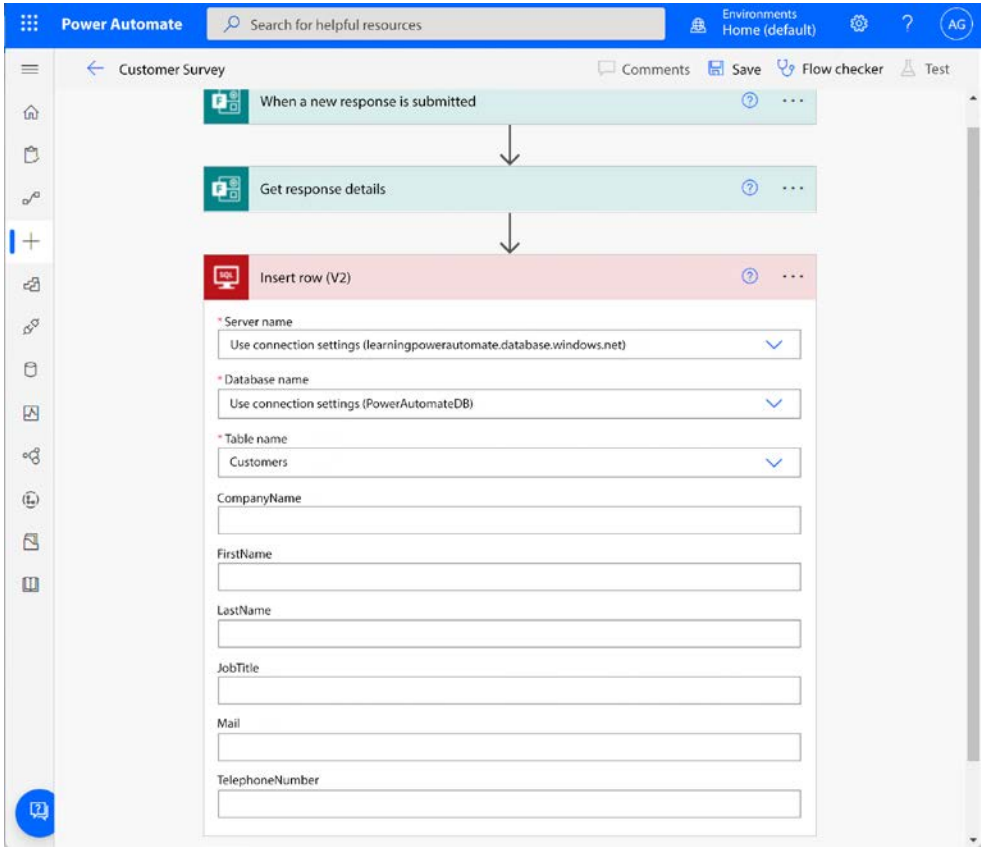


Figure 13.11: Configuring the Insert Row (V2) action

11. Select the **CompanyName** text box and then select the **CompanyName** dynamic content value from the **Get response details** action, as shown in *Figure 13.12*:

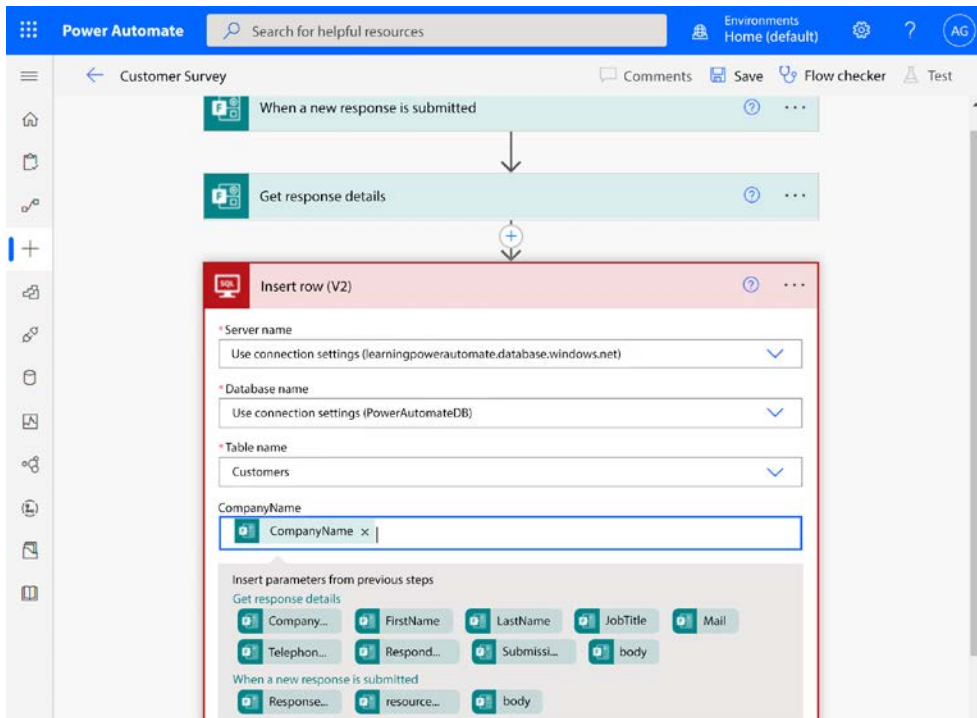


Figure 13.12: Adding the *CompanyName* dynamic content value

12. Repeat *step 11* for the remaining database column fields, matching them to the response values. Click **Save** when you've finished:

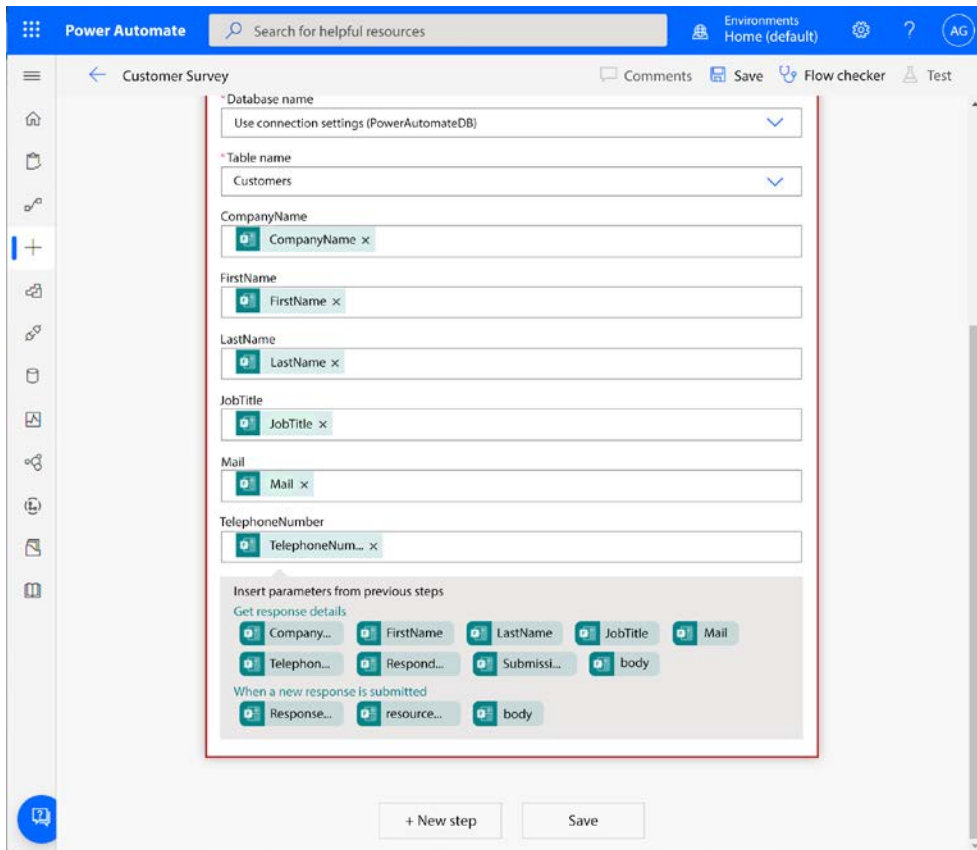


Figure 13.13: Finished placing all dynamic content values

The flow has been created. This flow will be triggered on a new form response and save the form data directly to a database. Next, we'll submit a form and then look for the results in the database.

## Testing the flow

In order to test the flow, you will need to submit a form response. To submit a form response, follow these steps:

1. Navigate to the Microsoft Forms dashboard (<https://forms.office.com>) and select your form under **My forms**:

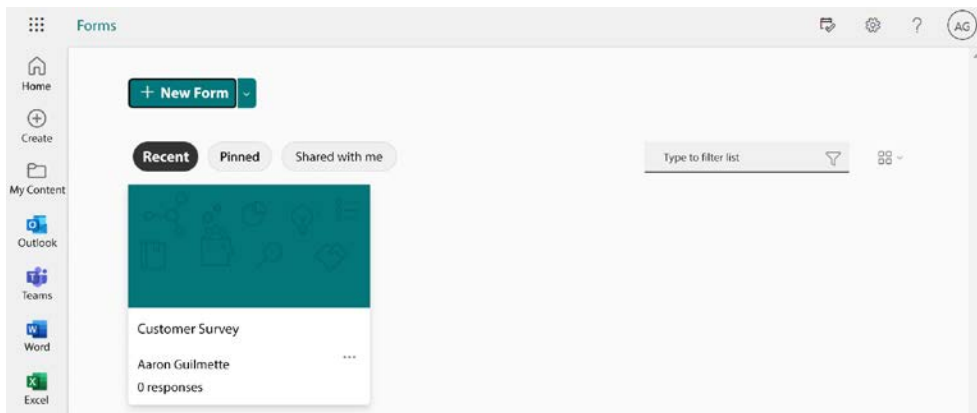


Figure 13.14: Forms dashboard

2. Select **Preview**.
3. Fill out the form and click **Submit**:

The screenshot shows a form submission screen. At the top, there is a 'Back' button and a toggle for 'Computer' and 'Mobile' views. The form contains four questions, each with a text input field: 3. LastName \* (value: Kol), 4. JobTitle \* (value: Marketing Manager), 5. Mail \* (value: aylak@contoso.com), and 6. TelephoneNumber \* (value: 4255551213). At the bottom, there is a 'Submit' button.

Figure 13.15: Submitting the form

After the form has been submitted, you will receive a confirmation page indicating that your submission was successful.

Next, we'll verify that the flow ran successfully.

## Verifying the result

Finally, you'll want to ensure that the flow has completed as you anticipated. You can check to see if the Forms data was written by flow to the database by examining the flow's run history or by querying the database table in SQL.

## Reviewing the run history

The easiest and quickest way to verify if the flow is successful is to examine the run history. The run history will show the steps performed during the flow's execution. You can review the run history for the flow by using the following process:

1. From the Power Automate web portal (<https://flow.microsoft.com>), click **My flows** and then select the **Customer Survey** flow.
2. Click the ellipsis for the flow and then select **Run history**.
3. Click on the date for the run:

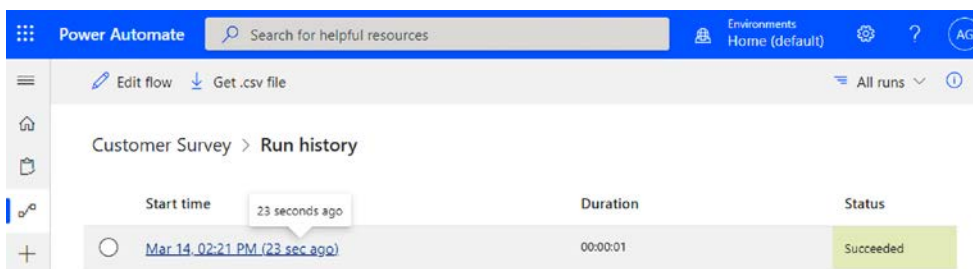


Figure 13.16: Expanding the flow run



4. Expand the **Insert row (V2)** action and review the data:

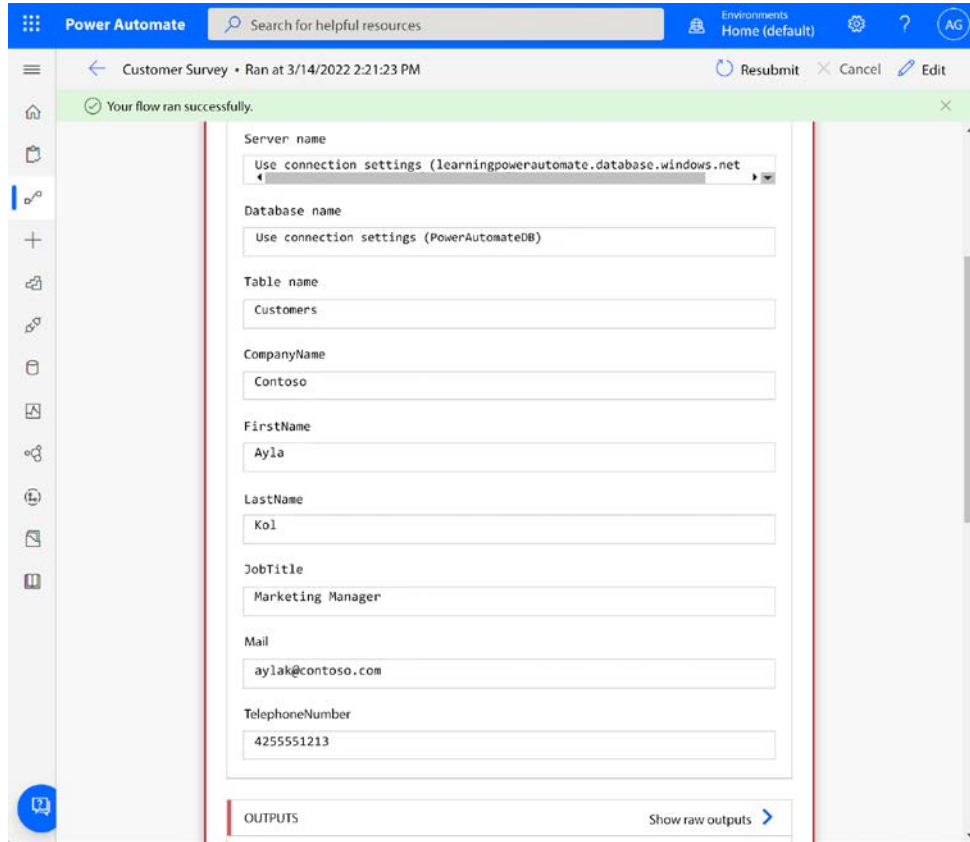


Figure 13.17: Viewing the run detail for the **Insert Row (V2)** SQL action

The expanded action will show the data that was processed from the form and inserted into the table.

## Reviewing the SQL data

You can also view the newly added data by using the SQL Query editor. To view the SQL data directly, follow these steps:

1. From the Azure portal (<https://portal.azure.com>), navigate to **SQL databases** and select the entry for the database you created in *Chapter 12, Using a Database*.
2. Select **Query editor (preview)**. If prompted, enter credentials to access the database and then click **OK**.

3. Copy and paste the following code into the Query editor:

```
SELECT * FROM [dbo].[Customers]
```

4. Click **Run**.
5. Review the data in the **Results** section, as shown in *Figure 13.18*:

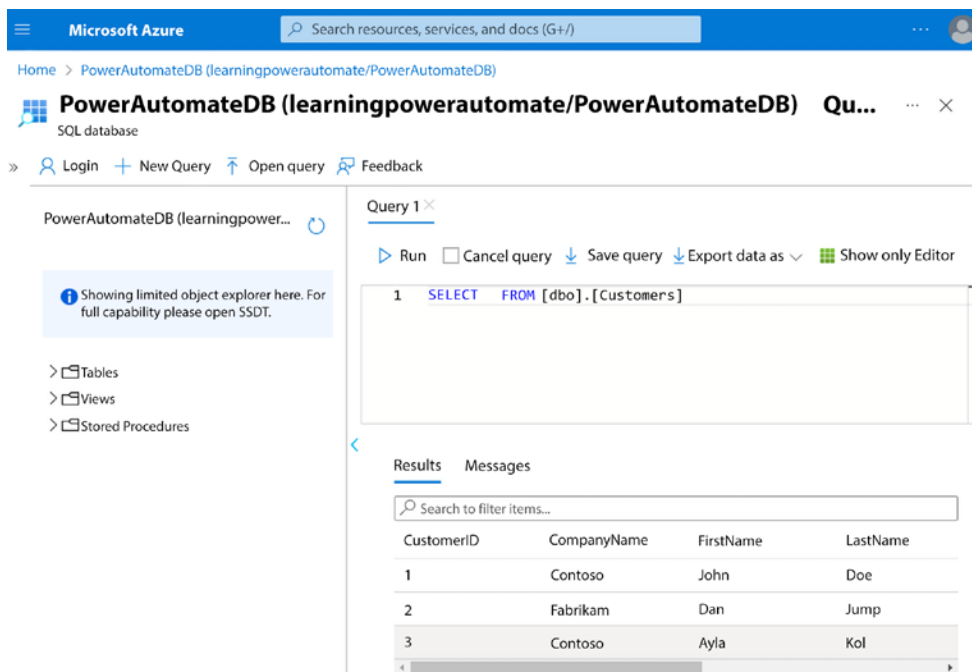


Figure 13.18: Reviewing the Results pane after executing a query

6. Compare the data in the **Results** section to the values you entered in the *Testing the flow* section.

No matter which way you choose to verify the results of the flow's execution, you should be able to see the same data. You can verify that the data in the table matches the data submitted in the form to confirm the flow is working correctly.

## Summary

In this chapter, you learned how to create a basic form using Microsoft Forms. Additionally, you were able to build upon the skills acquired in *Chapter 12, Using a Database*, to save data from Microsoft Forms into a SQL database.

In real-world scenarios, you could expand this flow further to generate a confirmation email back to the submitter or use a tool such as Excel or Power BI to sort the data. Depending on the type of data gathered in the form, you could even send this data to Azure Cognitive Services for sentiment analysis (which you'll learn about in *Chapter 17, Introducing AI Models*).

In the next chapter, we'll look at using Power Automate to gather user input directly. This will allow you to create more interactive flows.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 14

## Accepting User Input

Many types of business processing require some sort of user input. Input can come in the form of data typed in by a user, a choice prompt where the user selects “Yes” or “No,” or even uploading a file. Power Automate provides an interface for users to supply information as part of the flow. Connected applications, such as Power Apps, can also be used to supply information to a flow.

In this chapter, we’re going to revisit the concept of instant cloud flows from *Chapter 5, Creating Button Flows*. You’ll be introduced to gathering different types of user input that can be used later in a flow. Specifically, this chapter will cover the following topics:

- Understanding the user input options
- Creating a flow that uses all input types

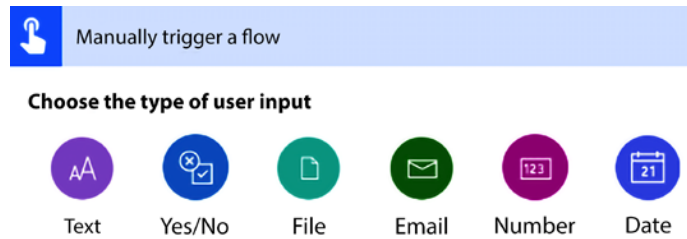
By completing the example exercise at the end of this chapter, you’ll become familiar with the different types of user input options and how they can be used to create interactive flows.

Onward!

### Understanding the user input options

User input can take many forms. When creating a manually triggered flow (also known as a *button* or *instant* flow), Power Automate allows flow designers to accept six types of user input. To add input capability to a manually triggered flow, click the title bar labeled **Manually trigger a flow**.

When the trigger object expands, click **Add an input** to be presented with the input options shown in *Figure 14.1*:



*Figure 14.1: Input data types*

The types of input are as follows:

- **Text:** The **Text** input option supports several configurations, including a free-form text field, a single drop-down selectable list, and a multi-select list.
- **Yes/No:** The **Yes/No** input option can be used to allow users to utilize a Yes/No toggle switch.
- **File:** The **File** input option provides the ability for the user to upload a file from their device. If the user is accessing the flow from a mobile device, they can also access the camera to capture and upload a picture.
- **Email:** The **Email** input option is designed to accept content formatted as an email address or allow the user to select from the organization's existing Exchange Online address book.
- **Number:** The **Number** input option allows users to enter numerals.
- **Date:** The **Date** option can be used to allow users to input a date, providing consistently and correctly formatted date values.

A manually triggered flow can use one or more input options and can accept multiple instances of the same input object type. For example, you may need to create a flow that has a series of Yes/No questions as well as an option to upload a file at the end.

All of the user input options can be configured as *required* or *optional*, providing a high degree of flexibility without the need for complex *if/then* coding statements.

Additionally, if your flow is capturing sensitive data (such as account credential data or other private values), you can configure the input for **Secure Inputs** and **Secure Outputs**. Select the ellipsis on the **Manually trigger a flow** title bar and select settings to review the action's options:

Settings for 'Manually trigger a flow'

Custom Tracking Id  
Set the tracking id for the run. For split-on this tracking id is for the initiating request.  
Tracking Id

Secure Inputs  
Secure inputs of the operation.  
Secure Inputs ☐ Off

Secure Outputs  
Secure outputs of the operation and references of output properties.  
Secure Outputs ☐ Off

Concurrency Control  
Limit number of concurrent runs of the flow, or leave it off to run as many as possible at the same time.  
Concurrency control changes the way new runs are queued. It cannot be undone once enabled.  
Limit ☐ Off

Trigger Conditions  
Specify one or more expressions which must be true for the trigger to fire.  
[+ Add](#)

Done Cancel

Figure 14.2: Configuring action options

**Secure Inputs** and **Secure Outputs** mask the details of the data displayed in a flow's run history (the actual user-submitted values are replaced with the text **Content not shown due to security configuration**), as shown in Figure 14.3:

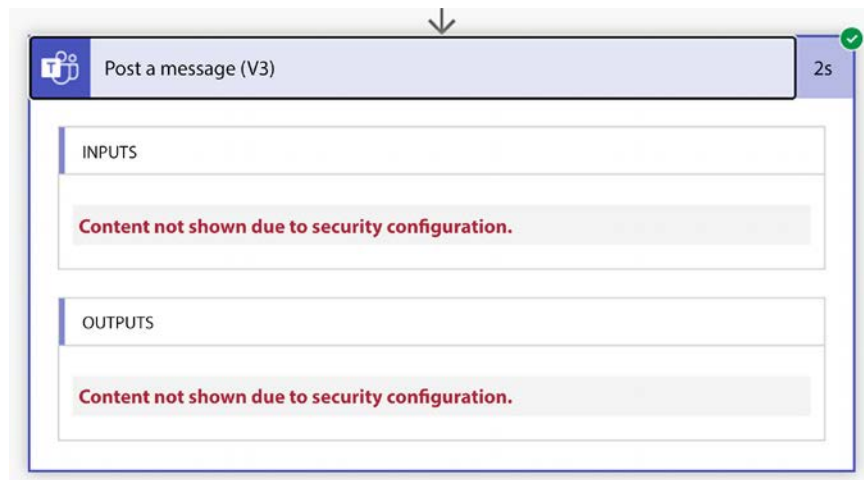


Figure 14.3: Viewing a flow's history where secure inputs and outputs have been configured

Now that you have a basic understanding of the types of input, let’s create a flow!

## Creating a flow that uses all input types

In this example, we’re going to configure a simple flow that uses all six input types and use that data to populate a SharePoint list. You could also try this example to save content to a database.

Before you begin configuring the flow, you’ll need to configure some basic prerequisites, such as a SharePoint list that contains the columns to store the various data values. Once the prerequisites have been completed, you’ll be able to move on to creating and testing the flow.

## Configuring the prerequisites

Since the data for this flow will be stored in a SharePoint list, you’ll need to configure a SharePoint list if you don’t already have one that you can use. In this example, we’ll create a new list in an existing site and then add one column to store each of our values.

To create the list, follow these steps:

- 1. Navigate to a SharePoint site, click **Site contents**, and then click **New**. Select **List** from the drop-down menu, as shown in *Figure 14.4*:

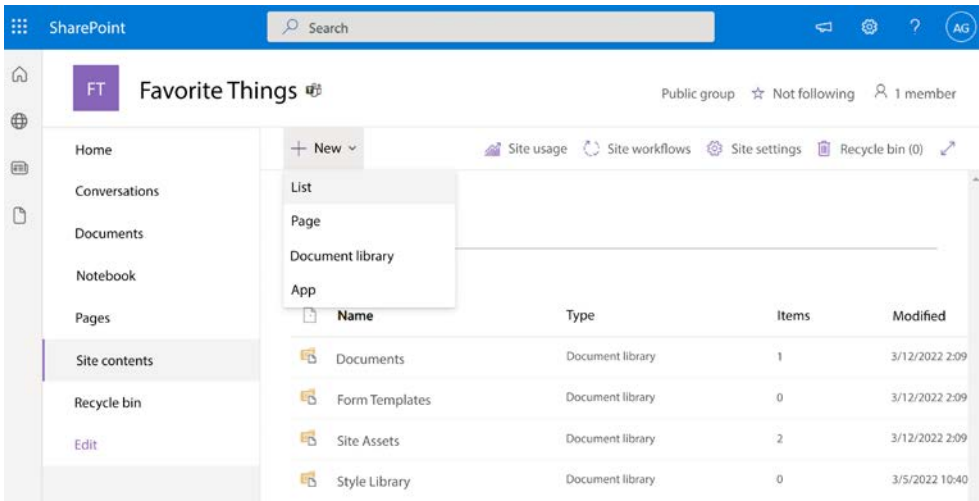


Figure 14.4: Creating a new list

- 2. Select **Blank list** from the template styles.
- 3. Enter a name for the list and click **Create**.

- 4. Click **Edit in grid view**:

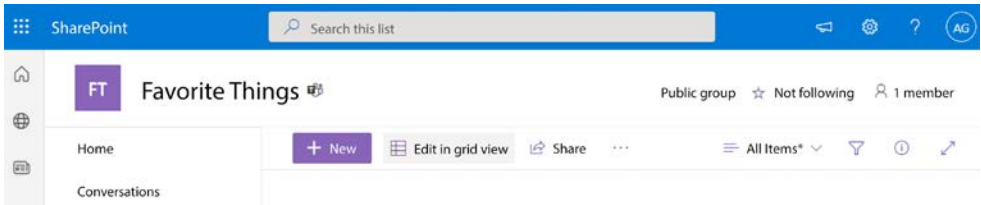


Figure 14.5: Launching Edit in grid view

- 5. Click the word **Title** and click **Rename Column**. Rename the column Name.
- 6. Click **Add column**. Select **Text** as the column type, as shown in Figure 14.6:

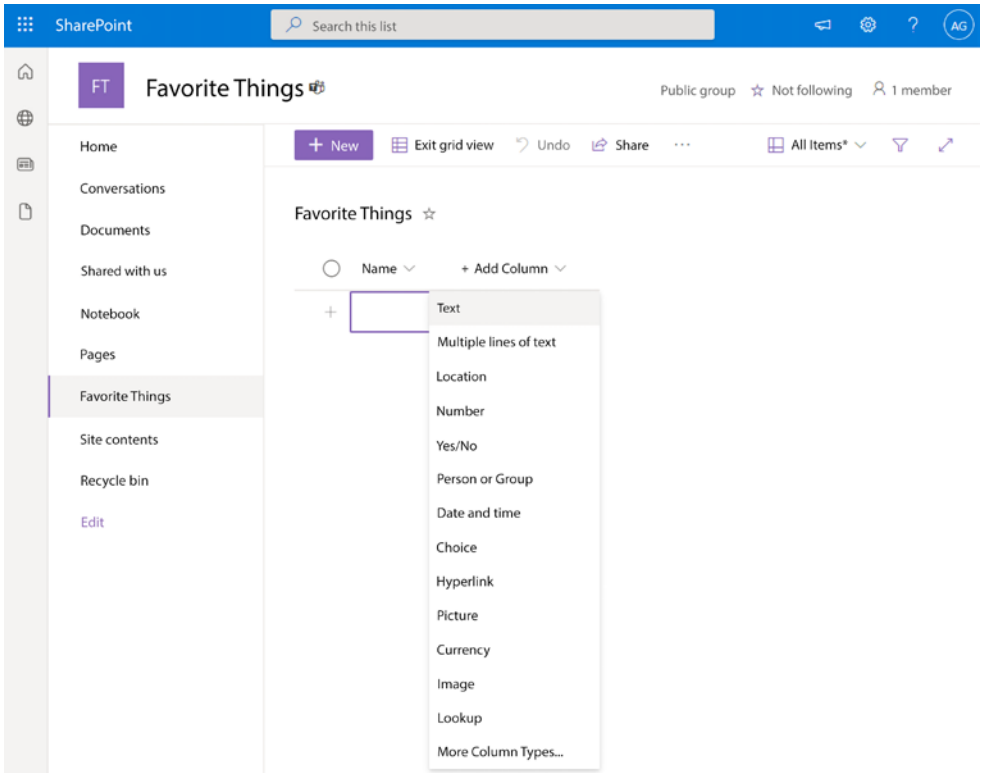


Figure 14.6: Creating a column and setting its type

- 7. In the **Create a column** fly-out, set the **Name** as Email Address, and ensure the **Type** dropdown is set to **Single line of text**. Click **Save**.



8. Repeat the procedure to add a column, using the following content types and values:

Type	Name
Text/Single line of Text	Email address
Yes/No	Eaten in the last week?
Number	How many times a month do you eat it?
Date	Today's date
Image	Picture of your favourite food

Table 14.1: List values

9. When you are finished, it should look similar to this:

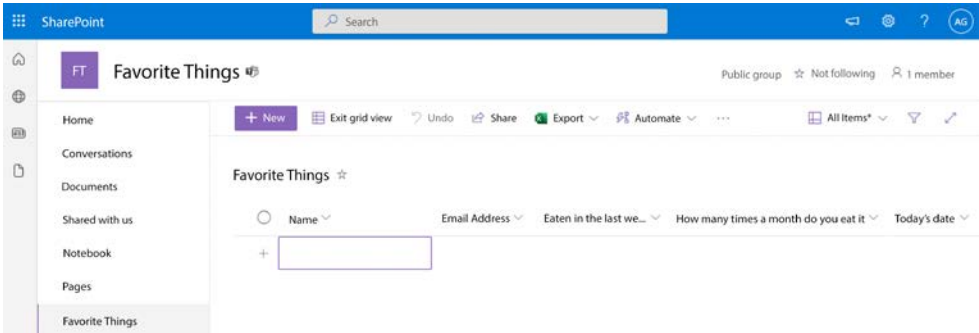


Figure 14.7: Configured list

10. Click **Exit grid view** to save your changes.

After a moment, your configured list should be saved. You’re ready to move on to the next step.

### Creating the flow

Now that you have a place to save the data, you can begin creating the flow. Follow these steps to create a manually triggered flow that captures user input:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>) and select **My flows**. Click **New flow**.
2. Select **Instant cloud flow**.
3. Enter a name for the flow, select the **Manually trigger a flow** trigger, and click **Create**.

4. Click the title bar for the trigger to display the **Add an input** option:

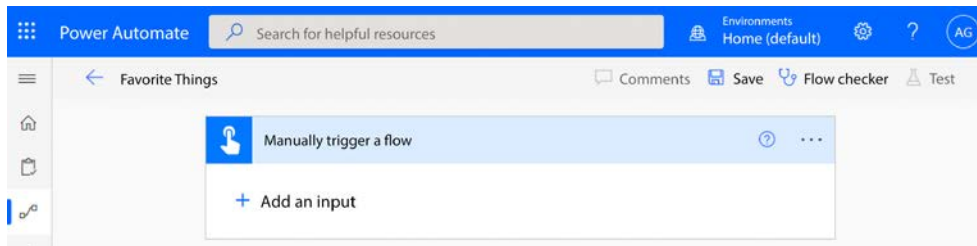


Figure 14.8: Expanding the trigger options

5. Click **Add an input** and select **Text**:

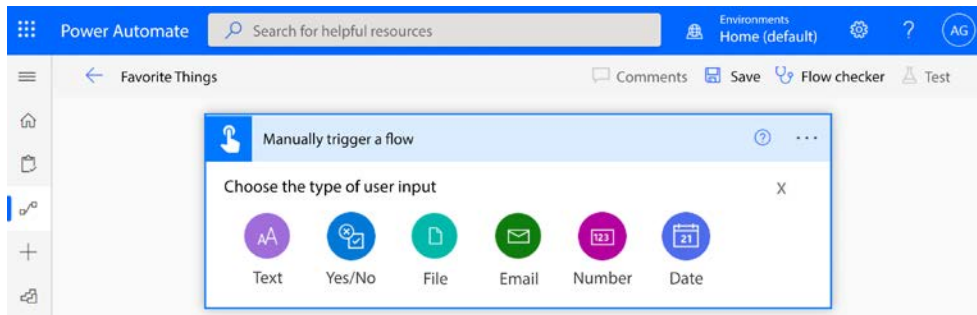


Figure 14.9: Adding a new input type

6. Update the label with the text **Name**, and then update the prompt to say **Please enter your name**:

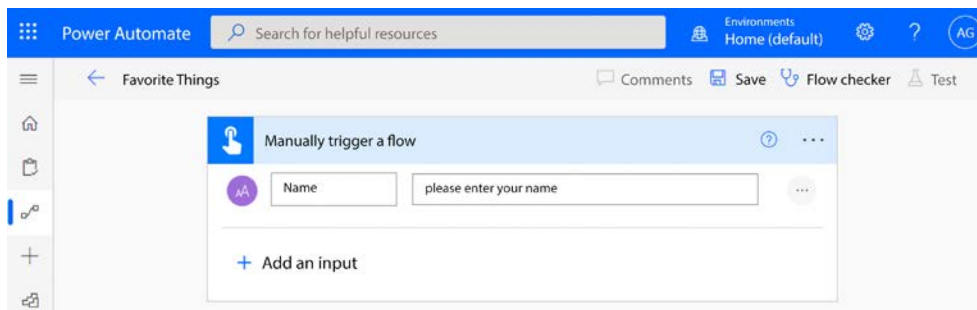


Figure 14.10: Configuring the text input

7. Click **Add an input** and select **Email**. Update the prompt to say **Please enter your e-mail address**.

8. Click **Add an input** and select **File**. Update the label to say **Picture**. Click the ellipsis (...) for the file item, and then toggle the option to **Make the field optional** or **Make the field required**, as desired:

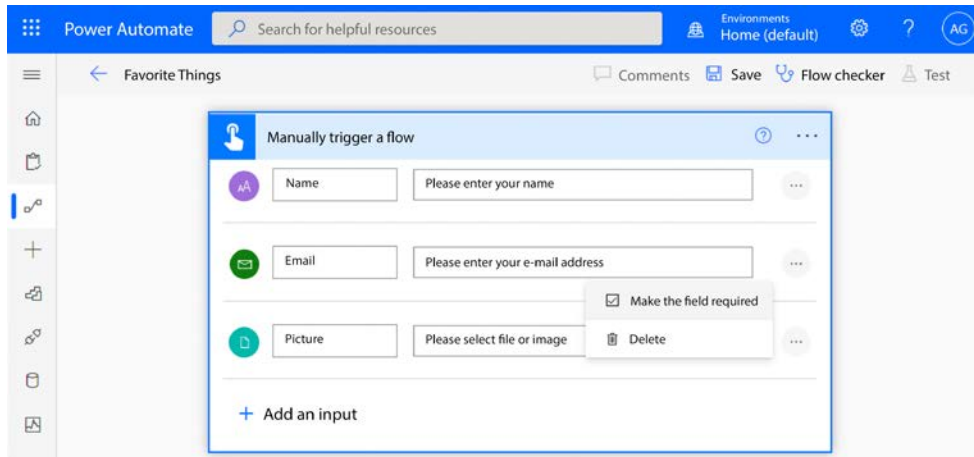


Figure 14.11: Configuring the picture input

9. Click **Add an input** and select **Yes/No**. Update the label to say **Have you eaten it in the last week?**
10. Click **Add an input** and select **Number**. Update the prompt to say **Please enter the number of times per month you normally eat it.**
11. Click **Add an input** and select **Date**. Update the label to say **Today's date**. Update the prompt to say **Please enter today's date.**
12. Click **New step**.
13. Select the SharePoint **Create item** action.

14. In the **Site Address** box, select the site containing the list you created for this exercise. If it is not displayed, click **Enter custom value** and add the URL for the SharePoint site.
15. Select the list. After a moment, the column names on the list should appear:

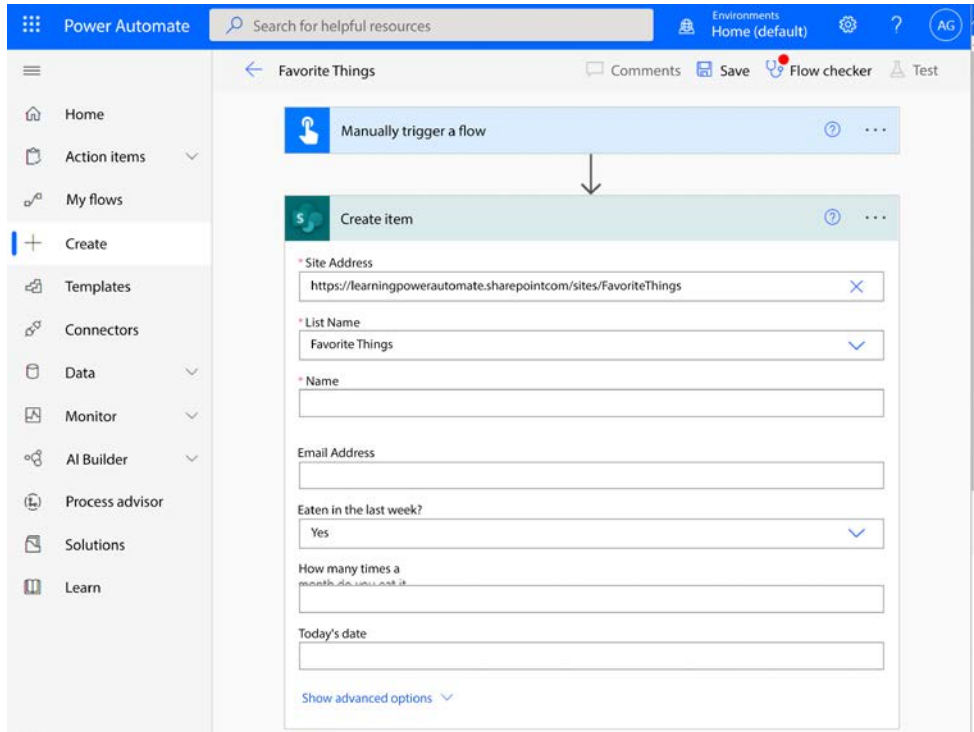


Figure 14.12: Editing the Create item action

16. Click the **Name** field, and then select the dynamic value token for **Name**.

17. Repeat the process for each of the remaining fields displayed in the **Create item** action card. For the **Eaten in the last week?** yes/no field, select **Custom value**, and then you'll be able to select the appropriate dynamic content token. When you've finished, it should look similar to the following screenshot:

The screenshot shows the Power Automate interface with the 'Create item' action card selected. The left sidebar contains navigation options: Home, Action items, My flows, Create (highlighted), Templates, Connectors, Data, Monitor, AI Builder, Process advisor, Solutions, and Learn. The main area displays the 'Favorite Things' flow configuration. The 'Create item' card is expanded, showing the following fields:

- Site Address:** `https://learningpowerautomate.sharepoint.com/sites/FavoriteThings`
- List Name:** `Favorite Things`
- Limit Columns by View:** `Avoid column threshold issues by only using columns defined in a view`
- Name:** `Name`
- Email Address:** `Email`
- Eaten in the last week?:** `Yes/No`
- How many times a month do you eat it?:** `Number`
- Today's date:** `Date`

At the bottom of the card, there is a link to 'Hide advanced options'.

Figure 14.13: Completing the Create item action

18. Click **New step**.
19. In the search box, type attachment. Select the **Add attachment** SharePoint action:

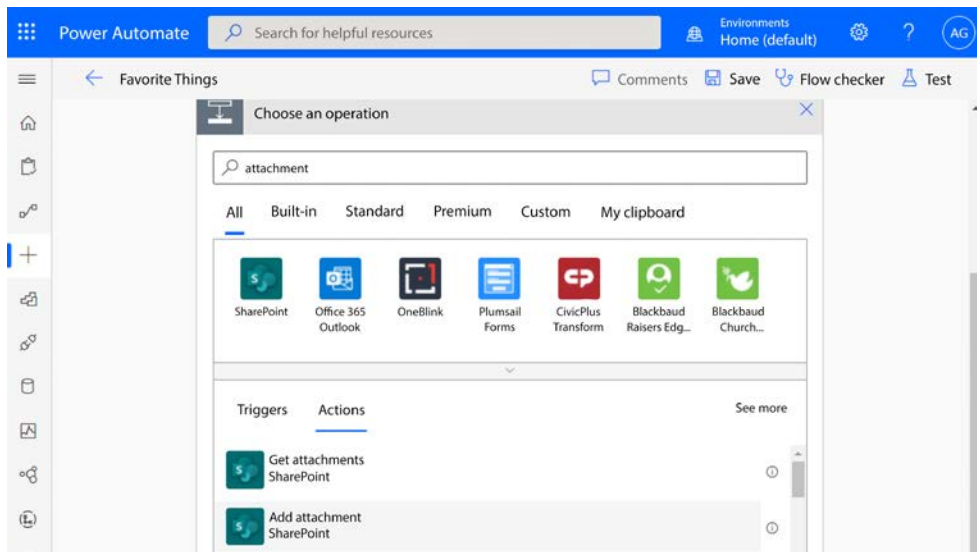


Figure 14.14: Selecting the Add attachment action

20. In the **Site Address** box, select the site containing the list. Select the list you used previously in the **List Name** box.
21. In the **Id** box, select the dynamic content token for **Id** under the **Create item** section.
22. In the **File Name** box, enter a filename to save the file as, such as `FavoriteThing.jpg`. You can also use an expression to dynamically name the item.

23. In the **File Content** box, select the **Picture** dynamic content token from the **Manually trigger a flow** section:

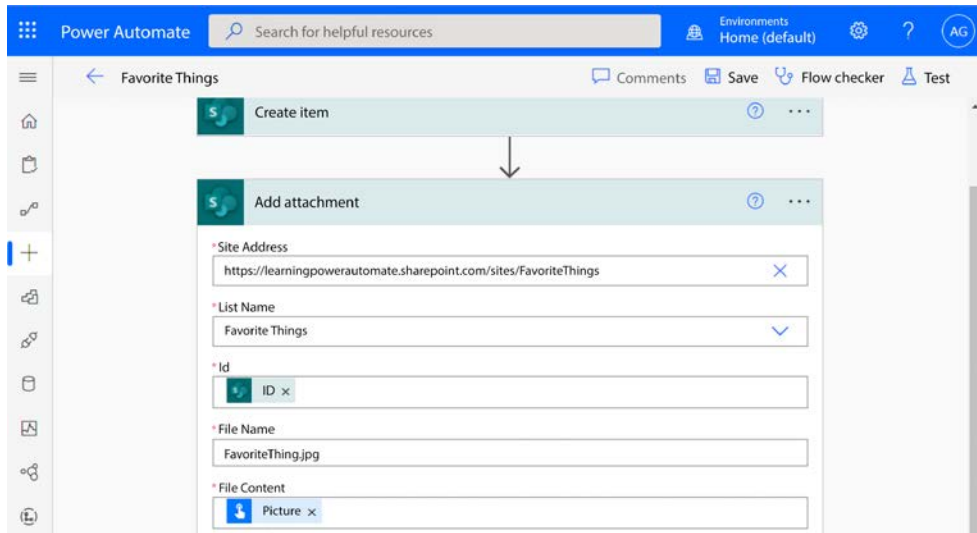


Figure 14.15: Adding the picture data to the SharePoint list item

24. Click **New step**. Select the **Send an email (V2)** Office 365 Outlook action.
25. In the **To** box, select the **Email** dynamic content token under the **Manually trigger a flow** action.
26. In the **Subject** box, enter a subject such as Thank you for your favorite food submission!
27. In the **Body** box, enter any text and select any dynamic content tokens you wish to include.
28. Click **Save**.

At this point, you have configured a flow that requests a number of input variables from a user and saves them to a SharePoint list. After creating the flow, you can click **Flow checker** to examine your flow for any errors.

Next, we'll test the flow. You can use either a mobile device with the Power Automate app installed or the web portal. In this example, we'll use a mobile device because we'll be selecting a picture from the device's camera roll.

## Executing the flow

To test this flow, we'll use a mobile device with the Power Automate app installed. To walk through the process, follow these steps:

1. On a mobile device, launch the **Power Automate** app.
2. Select **Buttons**.
3. Select the button for the new favorite food flow:

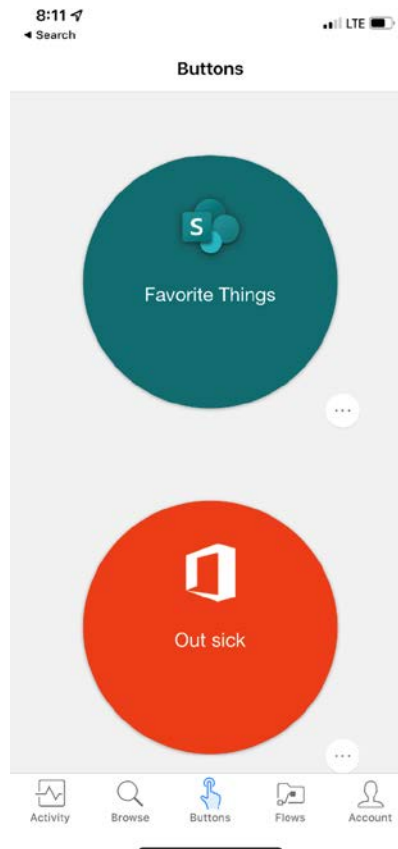


Figure 14.16: Launching the new button flow



## 4. Fill out the user input fields:

The screenshot shows a mobile application interface with a title bar at the top. The title bar contains the time '8:14', a search icon, the title 'Favorite Things', and a 'Done' button. Below the title bar is a list of input fields, each with a title and a placeholder text. The fields are: 'Name' (placeholder: 'Please enter your Name'), 'Email' (placeholder: 'Please enter your e-mail address'), 'Picture' (placeholder: 'Please select file or image'), 'Yes/No' (placeholder: 'Have you eaten it in the last week?'), 'Number' (placeholder: 'Please enter the number of times per month you normally eat it'), and 'Today's Date' (placeholder: 'Please enter today's date'). Each field is separated by a horizontal line. The 'Picture' field has a paperclip icon on the right. The 'Yes/No' field has a dropdown arrow on the right. The 'Number' field has a dropdown arrow on the right. The 'Today's Date' field has a dropdown arrow on the right. The bottom of the screen shows a home indicator bar.

8:14 Search

**Favorite Things** Done

**Name**  
Please enter your Name

**Email**  
Please enter your e-mail address

**Picture**  
Please select file or image

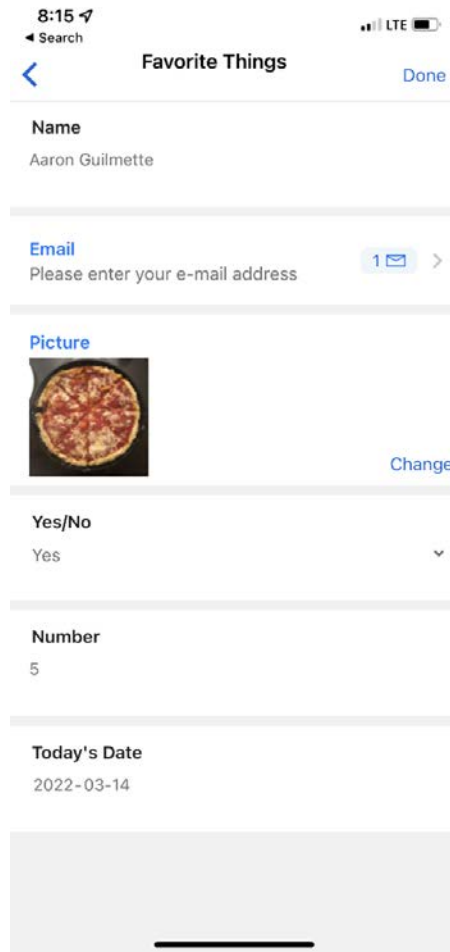
**Yes/No**  
Have you eaten it in the last week?

**Number**  
Please enter the number of times per month you normally eat it

**Today's Date**  
Please enter today's date

*Figure 14.17: Filling out the list items*

5. When you've finished, tap **Done**:



The screenshot shows a mobile app interface titled "Favorite Things". At the top, the status bar displays "8:15", "LTE", and a battery icon. Below the title bar, there is a search icon and a "Done" button. The form contains several fields: "Name" with the value "Aaron Guilmette", "Email" with a placeholder "Please enter your e-mail address" and a "1" icon, "Picture" with a pizza image and a "Change" button, "Yes/No" with a dropdown menu showing "Yes", "Number" with the value "5", and "Today's Date" with the value "2022-03-14". The bottom of the screen shows a home indicator bar.

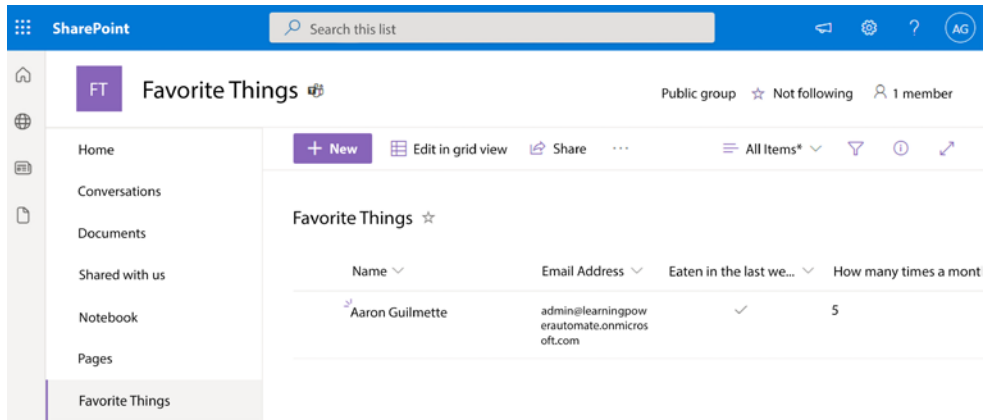
Figure 14.18: Completed data entry

The app will return you to the **Button flow** page. At this point, it's time to verify that the flow completed successfully.

## Verifying the flow execution

Once the flow has been submitted, you can review the data through any method you like:

- View the run history for the flow directly in the Power Automate mobile app by selecting **Flows** | the flow name | **Run history**
- View the run history for the flow from the Power Automate web portal (<https://flow.microsoft.com>) by selecting **My flows** | the flow name | **Run history**
- Browse to the SharePoint list, as shown in *Figure 14.19*:



*Figure 14.19: Reviewing the submitted data*

You can open an item's informational panel to view the detailed content and attachment data, as shown in *Figure 14.20*:

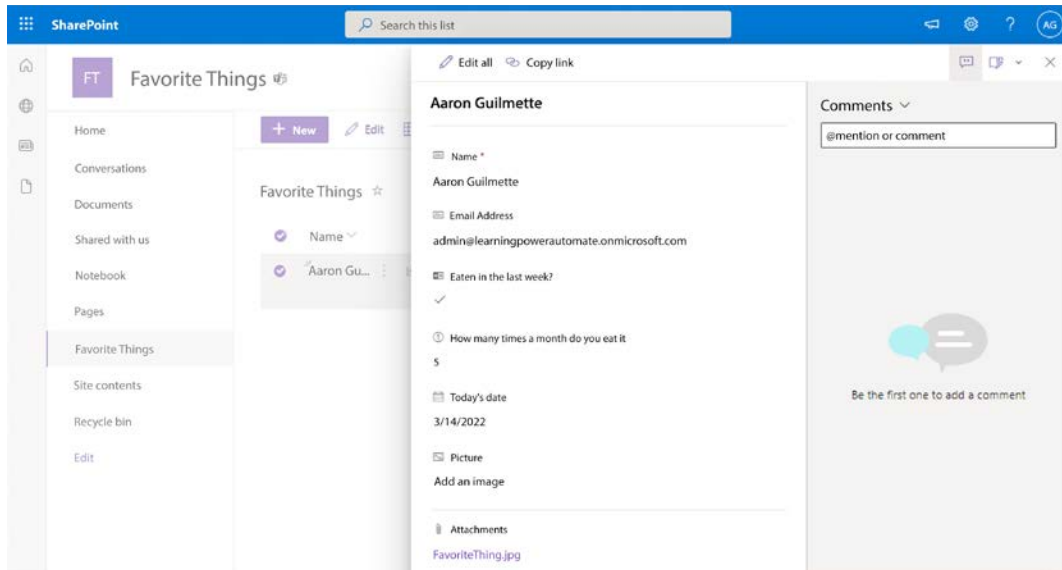


Figure 14.20: Reviewing the uploaded content on the SharePoint list

In this series of exercises, you created a SharePoint list to act as a repository, built a flow to capture a sample of all six user input types, tested it, and verified that the data was successfully uploaded.

## Expanding further

You may have noticed that the image shows up as an attachment inside the list item instead of in the picture column. At this time, the SharePoint **Create item** or **Update item** actions do not support manipulating an image column by default. With some creativity, however, you can craft a SharePoint HTTP request to upload the data directly.

Let's see how!

1. From **My flows**, select the **Favorite Things** flow for editing.
2. Insert a new step between the **Add attachment** and **Send an email** steps.

3. From the **Choose an operation** dialog, select **Compose**:

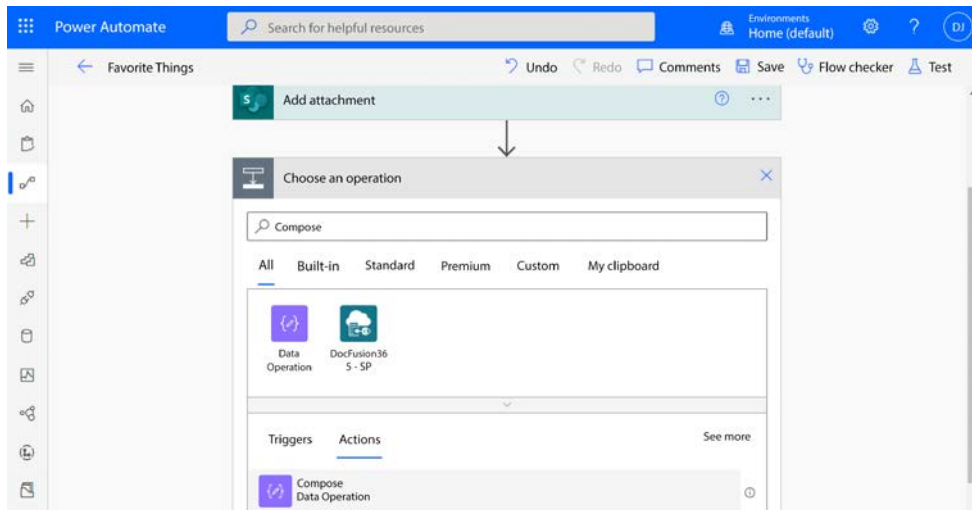


Figure 14.21: Adding a Compose action

4. In the **Inputs** area of the **Compose** action, enter the following data:

```
{
  "type": "thumbnail",
  "fileName": "FavoriteThing.jpg",
  "fieldName": "Picture",
  "serverUrl": "https://learningpowerautomate.sharepoint.com/sites/
FavoriteThings",
  "serverRelativeUrl": "@
{decodeUriComponent(decodeUriComponent(outputs('Add_
attachment')?['body/Id']))}",
  "id": "@{guid()}"
}
```

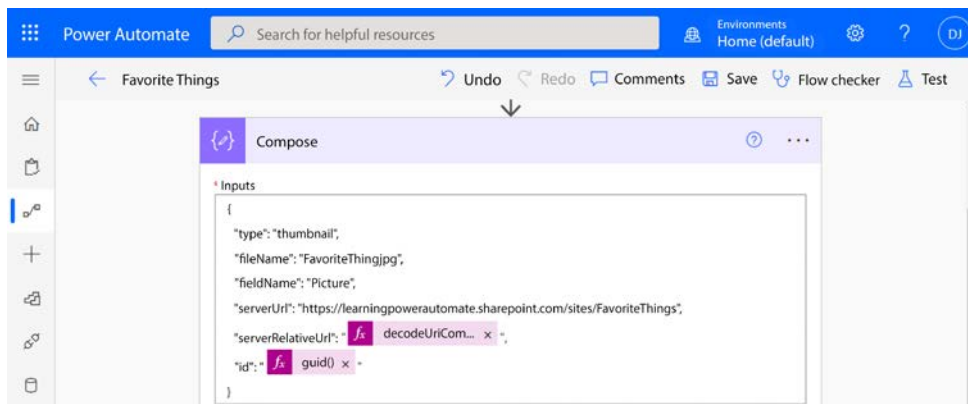


Figure 14.22: Configuring the Compose action

5. Insert a new step after the **Compose** action and before the **Send an email (V2)** action.
6. Select the **Send an HTTP request to SharePoint** action.
7. In the **Site Address** field, enter the site address containing the list.
8. Select **POST** as the **Method**.
9. In the **Uri** text field, enter `_api/web/lists/GetByTitle('Favorite%20Things')/items(@{outputs('Create_item')}['body/ID'])/ValidateUpdateListItem`, replacing the title of the list with your own list's name. If your list has a space in the name, you may need to replace the space character with its URL-encoded value, `%20`:

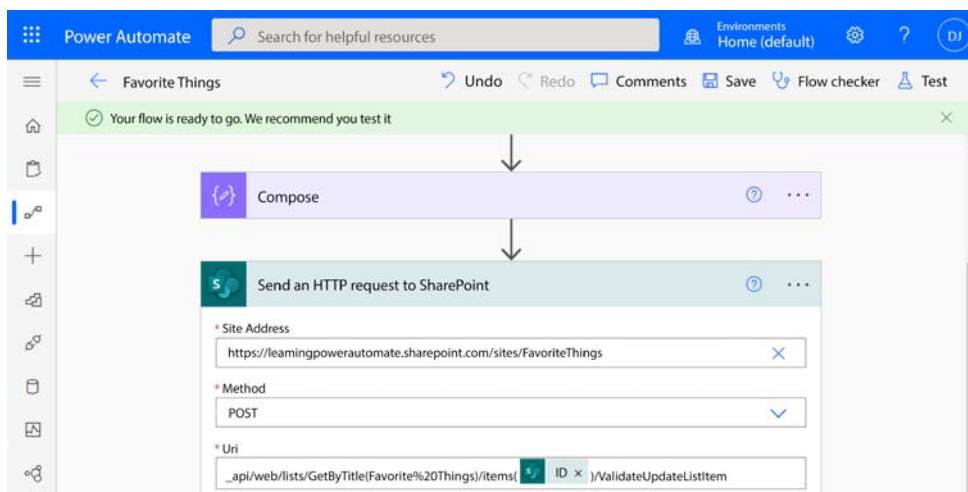


Figure 14.23: Configuring the HTTP request

10. In the **Headers** section, configure a header called Content-Type with the value application/json;odata=verbose.
11. Add a second header called Accept with the value application/json;odata=nometadata.
12. In the **Body** section, add the following content:

```
{
  "formValues": [
    {
      'FieldName': 'Picture',
      'FieldValue': '@{outputs('Compose')}'    }
  ]
}
```

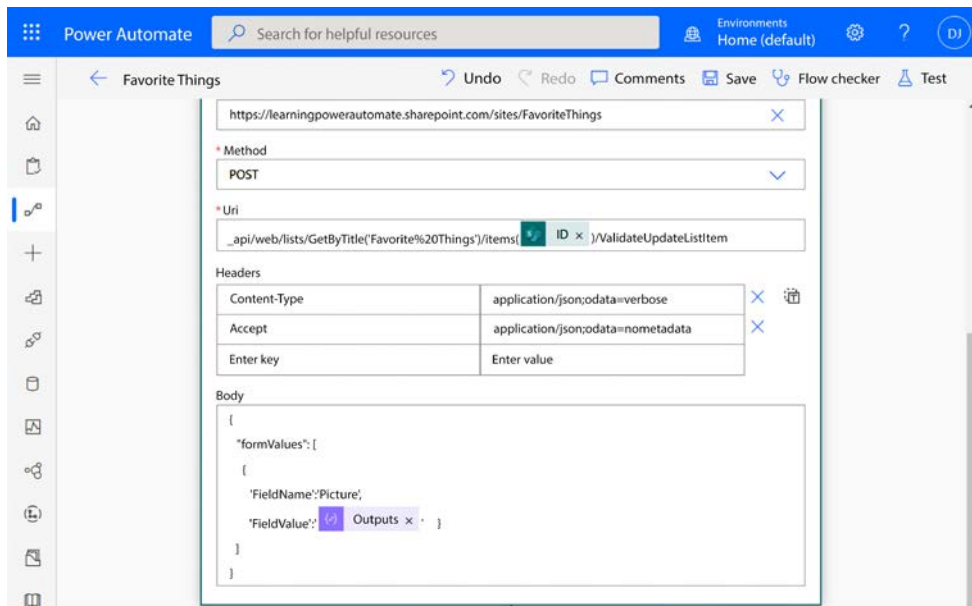
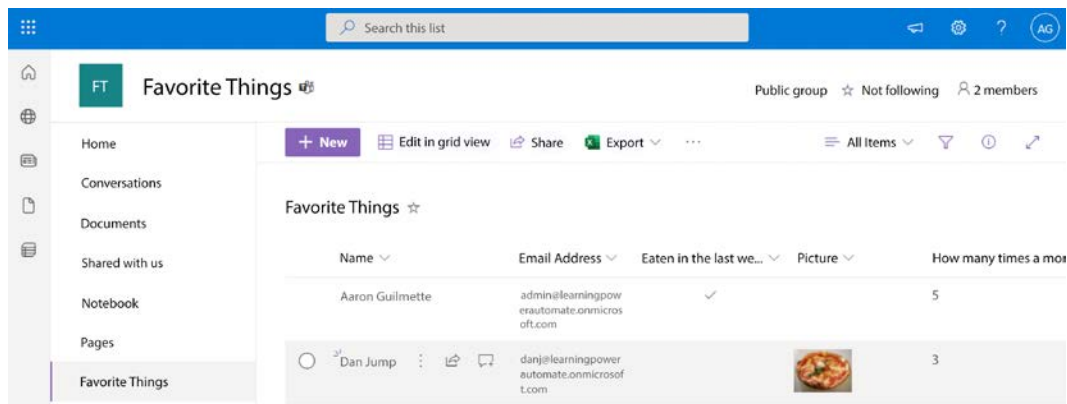


Figure 14.24: Configuring the HTTP request

13. Click **Save**.

This time, however, when you test your flow, the image should appear directly in the list, as shown in *Figure 14.25*:



*Figure 14.25: List showing a thumbnail of the uploaded image*

You can use this workaround to improve the quality of your list-based Power Automate flows. Adding images to SharePoint lists may take some additional work, but the result helps others visualize more data with fewer clicks.

## Summary

In this chapter, you learned about the six different kinds of user input that Power Automate can accept, including text, numbers, yes/no choice boxes, dates, and binary data. You also configured a flow that utilizes each of these methods and saved the content to a SharePoint list.

Requesting user input is a very useful feature of Power Automate, enabling a number of self-service and data gathering operations for your organization. With these types of input capabilities, you can enable workflow scenarios that rely on unique data provided at runtime, such as photos of objects or other types of content that can't be obtained without user interaction.

In the next chapter, we'll begin looking at advanced operational scenarios for Power Automate, such as interacting with Azure Active Directory.



## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 15

## Automating Azure AD

Up until this point, we've focused on using Power Automate as a workflow engine from a user's perspective. We've focused on processing email, SharePoint lists and documents, files and folders, and approval workflows.

Shifting gears away from purely business application automation, we're going to look at Power Automate's capabilities when it comes to working directly with **Azure Active Directory (Azure AD or AAD)** administration. Azure AD is the identity storage used for applications and services across the Microsoft 365 ecosystem. When it comes to reporting or group management there are a number of routine operational tasks that hold potential for automation. From an administrative perspective, they can be important tasks that come at a high labor cost if you need to continuously task switch between a ticketing system and one or more administrative interfaces.

While Azure AD does have a lot of native group management and reporting capabilities, there may be times when you need to create automations based on attributes that aren't available in the normal Azure AD management portal. While the data is present, it's not accessible in a way that's easily consumable from a native automation perspective.

Let's say you need to provide regular reporting to the security team on which users have not successfully completed **multifactor authentication (MFA)** registration. The data is available in Azure AD, but requires task switching to gather that data, process it, and then send an email. This type of activity seems to fall in the "important to be done, but not important for you to do it" category.

That's where this chapter comes in.

In order to get this advanced data, we're going to have to use some advanced skills, such as parsing JSON and working with the HTTP connector. We'll cover:

- Learning about connectors and actions
- Configuring prerequisites
- Creating an HTTP flow

By the end of this chapter, you should understand the basics of interacting with Azure AD using the HTTP connector.

Let's go!

## Technical requirements

Since we're going to be working with Azure AD to retrieve user data, you'll need an account that has global administrator access to configure your Azure AD and Microsoft 365 environments. If you're following along using a trial tenant that you created, you should already have all of the required permissions. However, if you're working on this in a production tenant where you don't have the required rights, you'll need to work with administrators who can configure the appropriate delegated permissions in the *Configuring prerequisites* section.



In addition to required permissions, you'll need to walk through a procedure to configure MFA in your tenant. Configuring MFA is outside the scope of this book. However, if you are interested in learning more about it or configuring it, you can visit <https://docs.microsoft.com/en-us/azure/active-directory/authentication/howto-mfa-getstarted>.

While MFA doesn't require Azure AD Premium, the reporting endpoint that this exercise utilizes does. You'll need at least one license of Azure AD Premium enabled in your tenant.

## Learning about connectors and actions

In this chapter, we're going to build on knowledge gained in other chapters, such as working with conditions, dynamic content tokens, and sending emails. We're going to then layer on several new concepts surrounding Azure AD, including the **HTTP**, **Parse JSON**, and **Create CSV table** actions.

### HTTP action

While working with Azure AD can typically be done using the standard Azure AD connector objects (such as the Azure AD connector and the Office 365 Groups connector), the property that stores MFA registration information is not exposed through those connectors.

In order to retrieve this data, we'll need to look to another source: the **Microsoft Graph API** (commonly referred to as just the **Graph API**). The Microsoft Graph API is a REST-based API that allows you to interact with data stored in Azure AD. Since the data we're looking for is stored in Azure AD, the Graph API is the natural choice for working with this data.

As you may recall from *Chapter 1, Introducing Microsoft Power Automate*, interacting with REST-based systems requires using HTTP to send verbs (sometimes also called actions, instructions, or methods) such as **GET** (read), **POST** (write), **PUT** (update), **PATCH** (partial update), and **DELETE** (remove).

To successfully use an HTTP action, we'll typically need to supply several pieces of information:

- **Method:** As mentioned earlier, we'll be choosing from GET, POST, PUT, PATCH, or DELETE.
- **URI:** The address used to contact the Graph API service (also called the **endpoint**).
- **Headers:** Sometimes optional, headers are information that describe what type of content the request is working with. Headers are frequently constructed in a key/value relationship, such as *Content-type: application/json*, where *Content-type* is the key and *application/json* is the value.
- **Queries:** Queries are parameters that you're sending to the service or endpoint. In the context of Azure AD, queries are frequently some sort of filtering criteria. Queries can be formatted in several ways and, similar to headers, can be structured in a key/value relationship.
- **Body:** The body contains the details of the request.

While some services and APIs are open and allow free querying of an endpoint, most will require some form of authentication to ensure that only authorized entities are accessing the data. In the case of working with Azure AD, we'll be specifying several pieces of information surrounding authentication.



You can experiment with other Microsoft Graph API actions using Microsoft Graph Explorer at <https://developer.microsoft.com/graph/graph-explorer>.

With other connectors and actions you've worked with so far, you have provided authentication by entering your username and password interactively while creating the flow. When working with HTTP requests and Azure AD, however, we'll need to specify that information separately, in the form of the *tenant ID*, *client ID*, and *client secret*.

## Parse JSON

As mentioned in *Chapter 1, Introducing Microsoft Power Automate*, connectors store information in **JavaScript Object Notation** format, or **JSON**. When working with REST-based APIs, the data retrieved will frequently be formatted as JSON.

In order to extract the information we need to create the report, we'll need to use the **Parse JSON** action, which will require us to configure Power Automate to understand the **schema** being used. A schema is a type of definition or legend used to describe the structure of the JSON data. We'll use this to help identify the fields or properties that we need to build the report.

## Create CSV table

In this example, we'll use a **comma-separated value (CSV)** table to store the output. The **Create CSV table** action will be used to compile information into a simple spreadsheet-style report that can be attached to an email.

Now that you understand a little bit more about the actions that will be used, let's start setting up the environment to support the flow.

## Configuring prerequisites

Since the HTTP flow that we're going to configure doesn't have a way to enter a traditional username and password as the authentication mechanism, we're going to create or register a special application with the appropriate permissions.

In addition, you'll also need to configure a **client secret**, which is the application's equivalent to a password. Together, the tenant information, application information, and client secret will be used to provide the necessary authentication to execute the flow.

## Configuring an Azure AD application

To register a new Azure AD application for this flow, follow these steps:

1. Log in to the Azure AD portal (<https://portal.azure.com>) using an identity with global administrator privileges.

2. In the search bar, search for App registrations and select it:

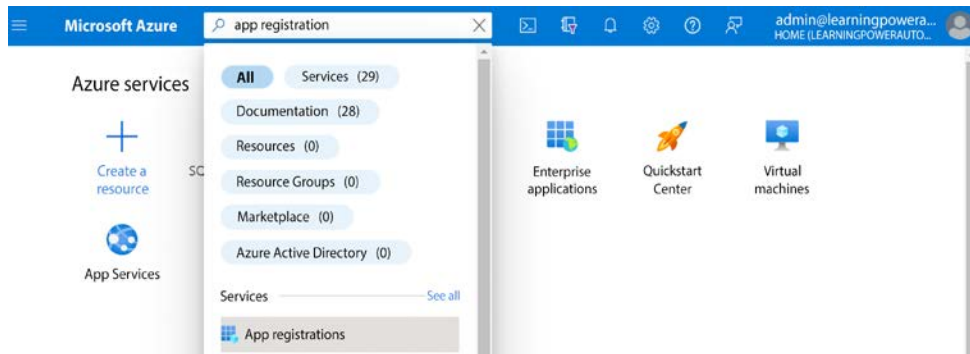


Figure 15.1: Navigating the Azure portal

3. Click **New registration**:

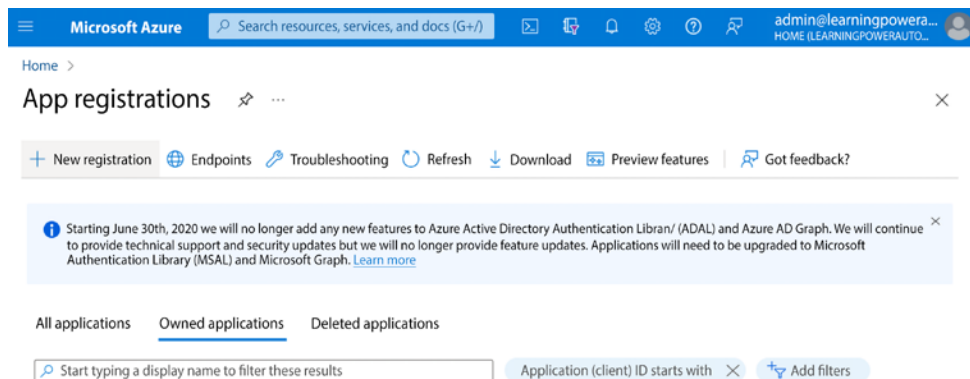



Figure 15.2: Creating a new app registration

4. Configure a **Display name** for the application.

5. Select **Accounts in this organizational directory only (Home only – Single tenant)** under **Supported account types**, and then enter `https://auth` as the **Redirect URI**, as shown in *Figure 15.3*. Click **Register** when complete:



It's best to create a separate registration for every application or use case and not to reuse the app registration. This allows you to scope the permissions more closely for the particular requirement.

Microsoft Azure

Search resources, services, and docs (G+)

admin@learningpowera...  
HOME (LEARNINGPOWERAUTO...

Home > App registrations >

Register an application

\* Name

The user-facing display name for this application (this can be changed later).

Power Automate MFA Report

Supported account types

Who can use this application or access this API?

☒ Accounts in this organizational directory only (Home only - Single tenant)

☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant)

☐ Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

☐ Personal Microsoft accounts only

[Help me choose...](#)

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Public client/native (mobile ... `https://auth`

By proceeding, you agree to the [Microsoft Platform Policies](#)

Register

Figure 15.3: Configuring the application registration

6. Under **Manage**, select **API permissions**:

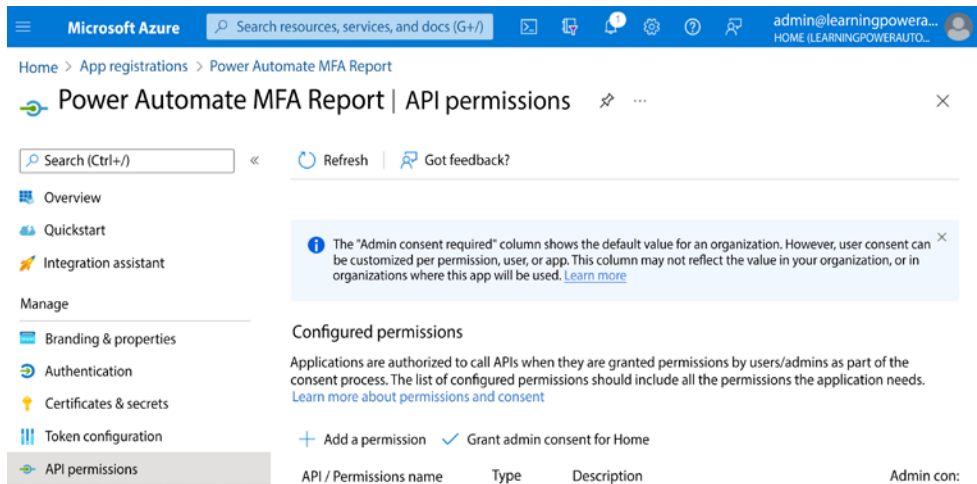


Figure 15.4: Navigating to API permissions

7. Click **Add a permission**.
8. On the **Select an API** page, choose **Microsoft Graph**:

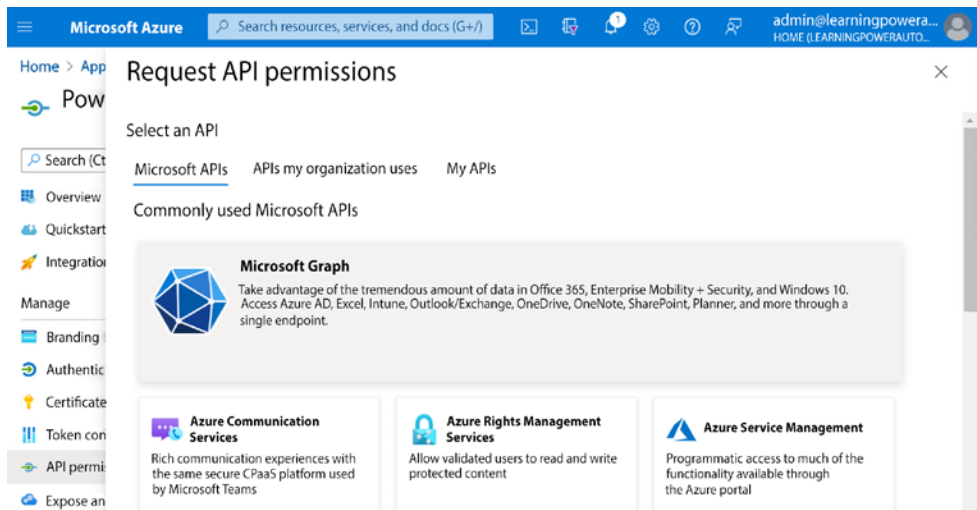


Figure 15.5: Selecting an API



9. Select **Delegated permissions**. On the **Select permissions** pane, select **Reports.Read.All** and click **Add permissions**:

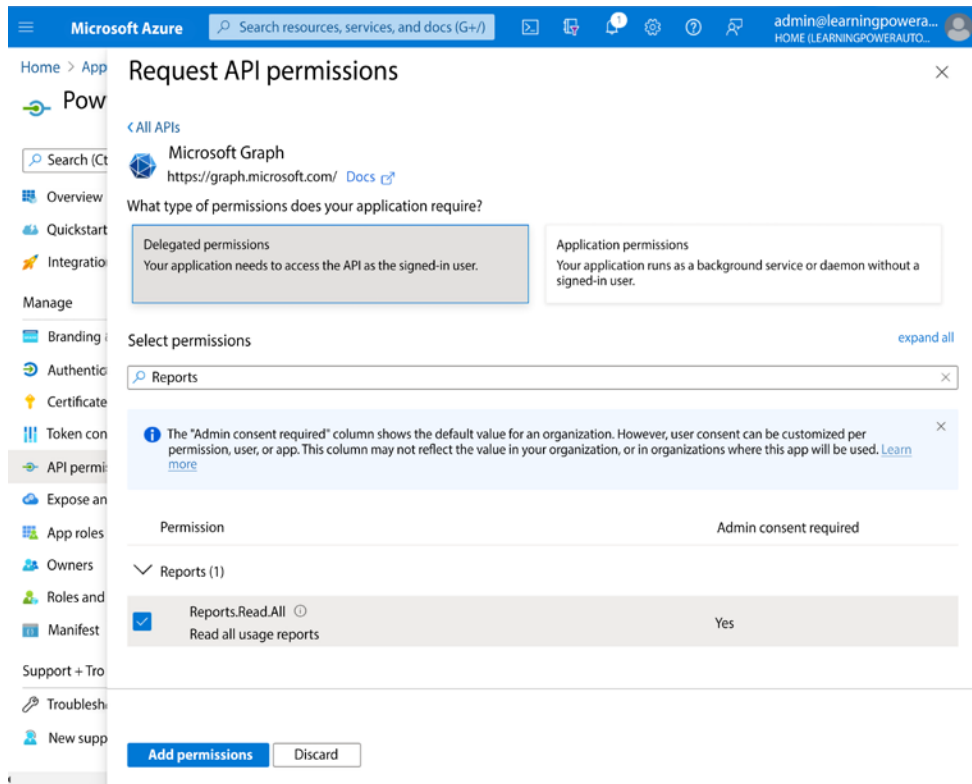


Figure 15.6: Adding delegated API permissions

10. Click **Add a permission**.
11. On the **Select an API** page, choose **Microsoft Graph**.
12. Select **Application permissions**. On the **Select permissions** pane, select **Reports.Read.All** and click **Add permissions**.
13. Click **Add a permission**.
14. On the **Select an API** page, choose **Microsoft Graph**.
15. Select **Application permissions**. On the **Select permissions** pane, select **AuditLog.Read.All** and click **Add permissions**.
16. Click **Add a permission**.
17. On the **Select an API** page, choose **Microsoft Graph**.

18. Select **Delegated permissions**. On the **Select permissions** pane, select **AuditLog.Read.All** and click **Add permissions**.



The need for the **AuditLog.Read.All** permissions is currently filed as a bug. While the documentation indicates this permission is not necessary, failing to configure this permission may occasionally result in the following error: *Error details: Application missing MSGraph API 'Read all audit log data' permission*. Adding the **AuditLog.Read.All** permission will resolve this issue.

19. Click **Add a permission**.
20. On the **Select an API** page, choose **Microsoft Graph**.
21. Select **Application permissions**. On the **Select permissions** pane, select **Directory.Read.All** and click **Add permissions**.
22. Click **Add a permission**.
23. On the **Select an API** page, choose **Microsoft Graph**.
24. Select **Delegated permissions**. On the **Select permissions** pane, select **Directory.Read.All** and click **Add permissions**.

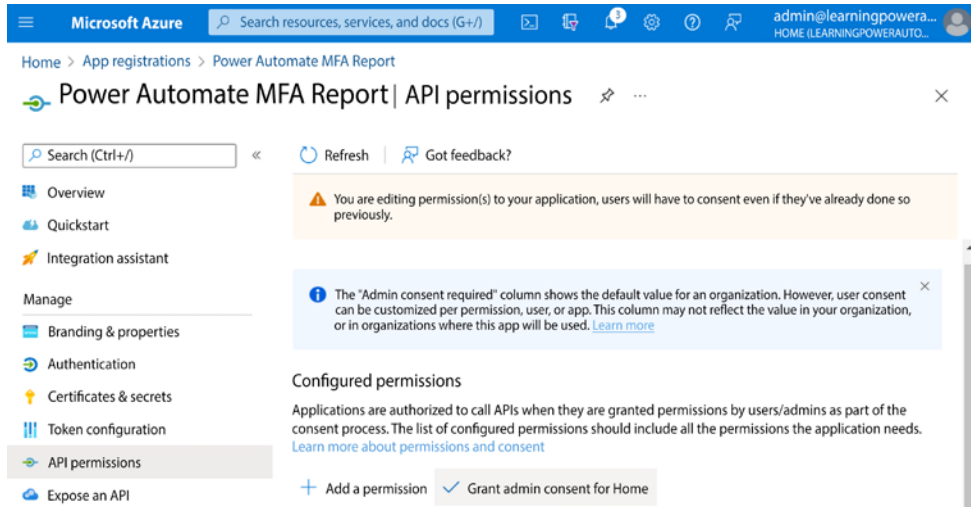


The **Directory.Read.All** permissions should only be necessary if the cache for the licenses in a tenant has not been updated recently (usually hourly). If you are attempting to run the flow shortly after adding premium licensing to your tenant, you may need to add this permission temporarily to allow the registered application the ability to gather licensing data from the tenant.

If you do not have an Azure AD Premium subscription enabled in your tenant, you may receive the following error: *Error Details: Neither tenant is B2C or tenant doesn't have premium license*. To resolve this error, you will need to either start a trial or purchase a minimum of one Azure AD Premium P1 license.

25. Click **Add a permission**.
26. On the **Select an API** page, choose **Microsoft Graph**.
27. Select **Application permissions**. On the **Select permissions** pane, select **UserAuthenticationMethod.Read.All** and click **Add permissions**.
28. Click **Add a permission**.
29. On the **Select an API** page, choose **Microsoft Graph**.

30. Select **Delegated permissions**. On the **Select permissions** pane, select **UserAuthenticationMethod.Read.All** and click **Add permissions**.
31. On the **API permissions** page, click **Grant admin consent**, as shown in *Figure 15.7*:



*Figure 15.7: Granting admin consent*

32. Click **Yes** to approve the admin consent operation.

With the application created, let's move on to creating a client secret.

## Creating a client secret

In this section, you'll be creating a client secret and capturing the application and tenant details. This information is going to be used as the authentication data for the HTTP flow.

To create the client secret and save the necessary information, follow these steps:

1. With the app registration page still open, select **Certificates & secrets** in the **Manage** section:

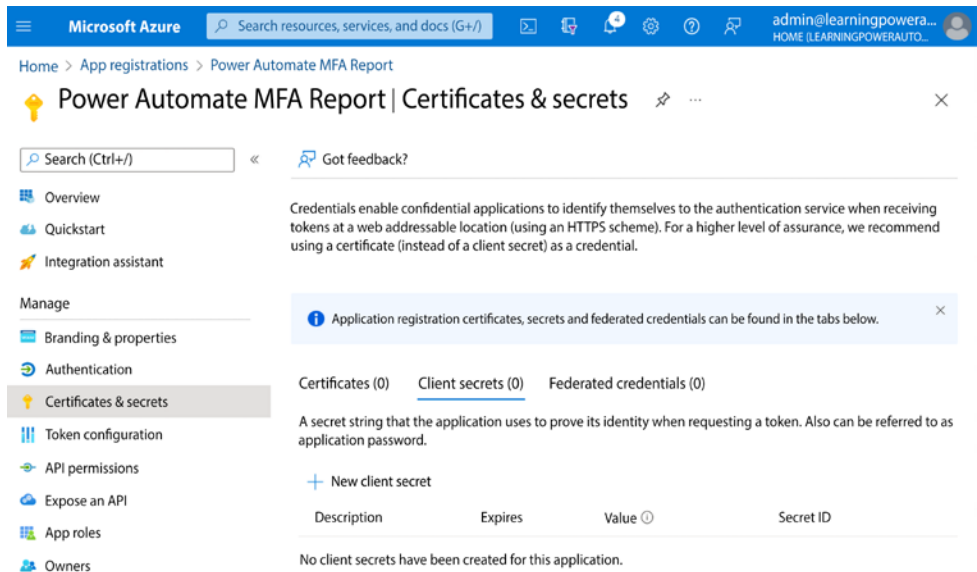


Figure 15.8: Certificates & secrets

2. Select **New client secret**.
3. On the **Add a client secret** page, enter a **Description** and chose a setting from the **Expires** dropdown to configure an expiration date. Click **Add** when you're finished:

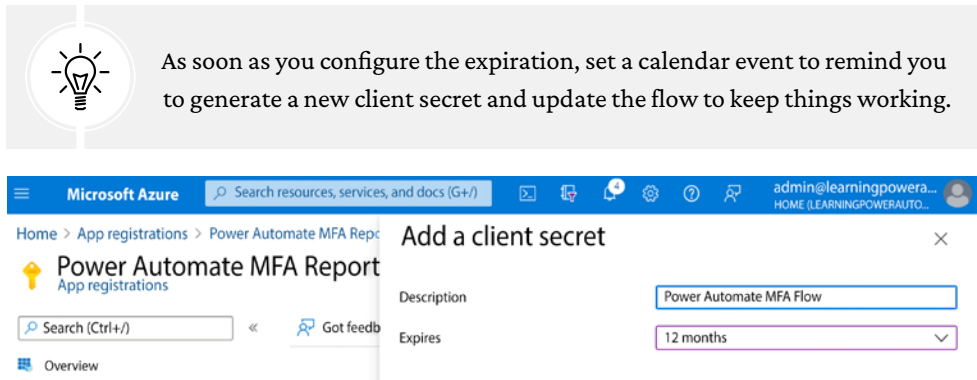


Figure 15.9: Configuring the client secret

4. As soon as you click **Add**, you'll have access to the value of the client secret. You'll need to copy and save this value immediately. As soon as you navigate away from this page, you'll be unable to retrieve the value and will have to create a new one. Click the **Copy** link to copy the data to the clipboard; paste it into a text document for safekeeping:

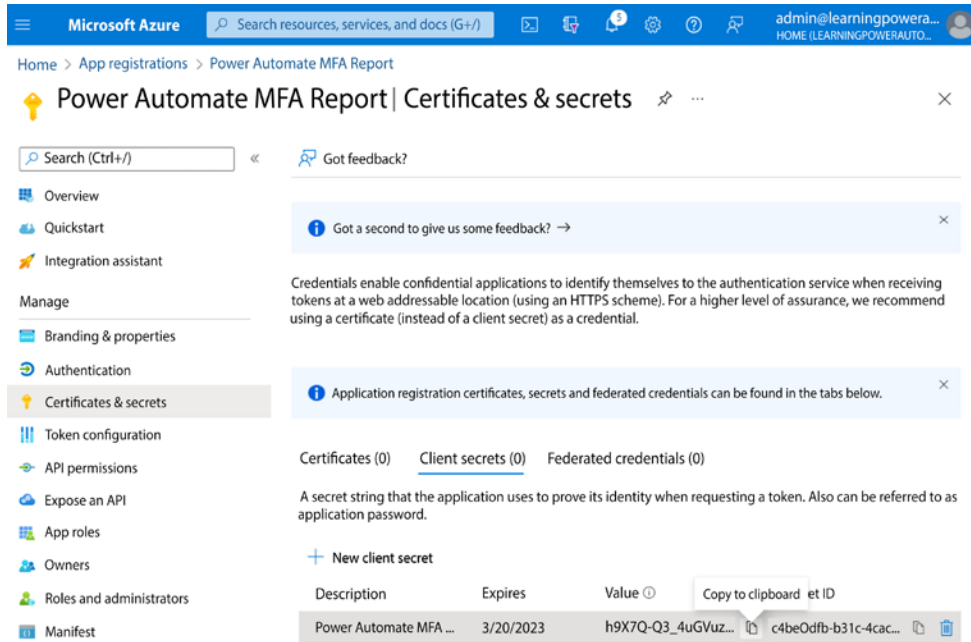


Figure 15.10: Capturing the client secret details

5. On the navigation pane, select **Overview**.
6. Capture the **Application (client ID)** and **Directory (tenant) ID** values and save them to the same text file you created in *step 4*:

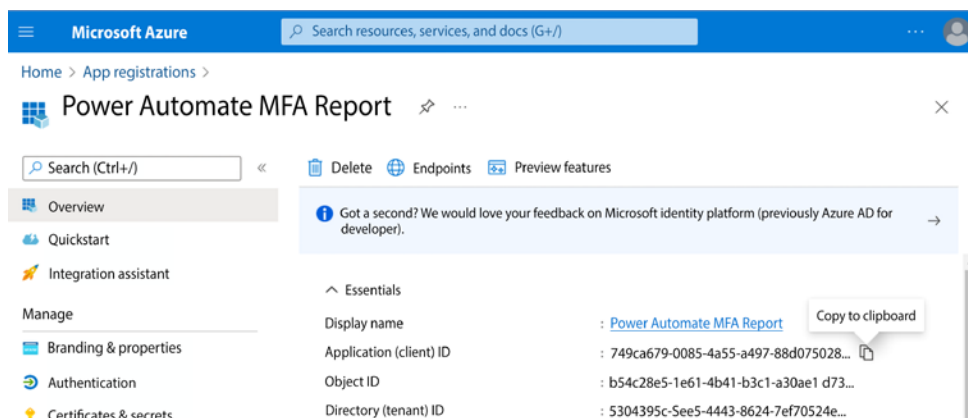


Figure 15.11: Capturing additional configuration information

That's it!

With the prerequisites configured, we can move on to creating the flow itself.

## Creating an HTTP flow

In this section, we're going to create the flow to retrieve the MFA registration data. Let's begin!

### Gathering the data

The first part of creating the flow is going to require connecting to Azure AD to retrieve the data necessary for building a report. We'll use an HTTP action to query Azure AD.



In this chapter, we're just going to focus on one of the REST resources in the Microsoft Graph API endpoint. There are hundreds of different resources that you can work with to retrieve and modify data. For a complete list of Graph API resources, see <https://docs.microsoft.com/en-us/graph/api/overview>.

To configure the HTTP action, follow these steps:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>) and click **Create**.
2. Click **Scheduled cloud flow**.
3. Configure a **Flow name**.
4. Configure the schedule settings under **Run this flow** to execute weekly, and click **Create** when finished:

The screenshot shows the 'Build a scheduled cloud flow' dialog box. It includes an illustration of a laptop with a clock icon. The 'Flow name' field contains 'Generate Non-MFA Report'. Under 'Run this flow', the 'Starting' date is '3/20/22' at '10:00 AM'. The 'Repeat every' setting is '1 Week'. The 'On these days' section shows the days of the week with 'M' (Monday) selected. A summary line states 'This flow will run: On Monday every week'. At the bottom right, there are three buttons: 'Skip', 'Create' (which is highlighted in blue), and 'Cancel'.

Figure 15.12: Configuring the flow schedule settings

5. Click **New step**.
6. Select the **HTTP** action:

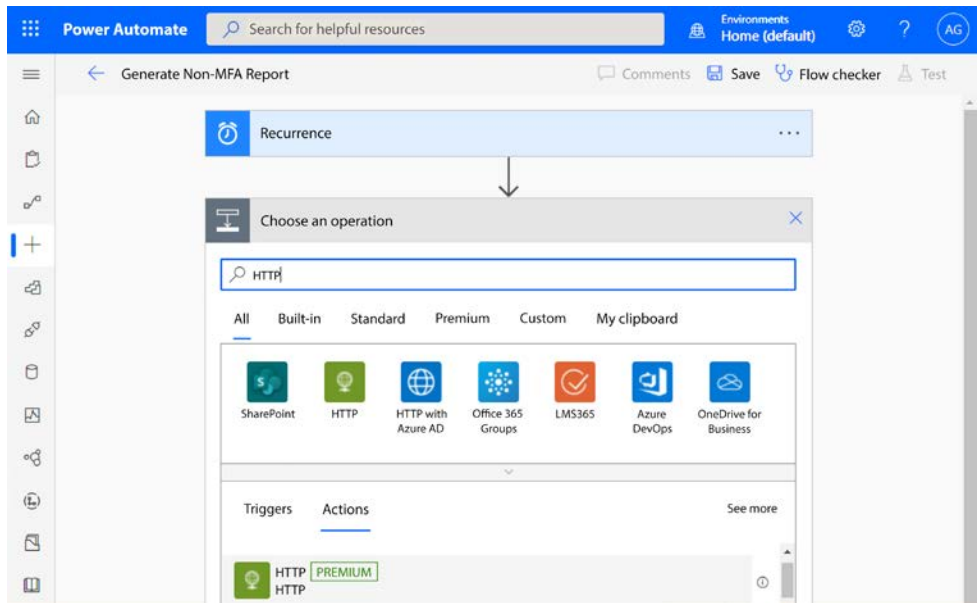


Figure 15.13: Selecting the HTTP action

7. In the **HTTP** action, under **Method**, choose **GET**:

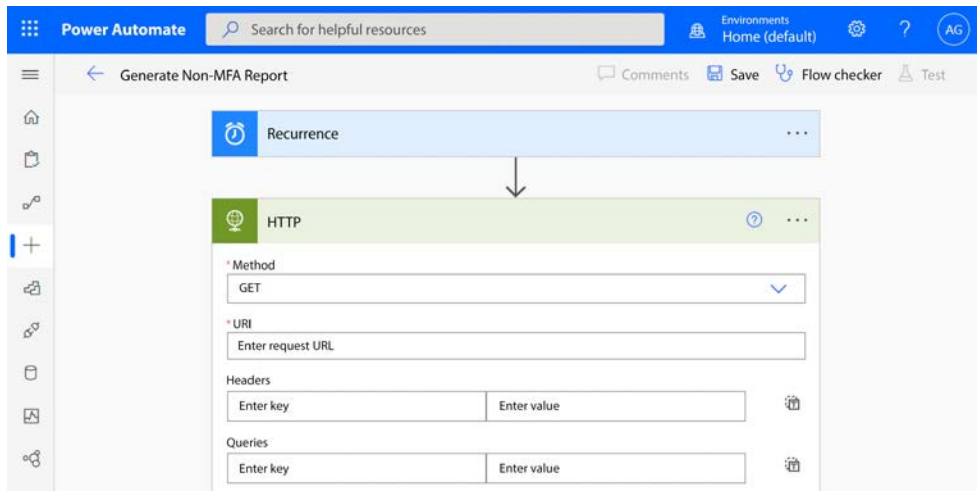


Figure 15.14: Configuring the HTTP action



- 8. Click **Show advanced options** at the bottom of the HTTP action to expose the additional parameter fields.
- 9. Continue filling out the rest of the HTTP action parameters, using the values in *Table 15.1* as a guide. Populate the **Tenant**, **Client ID**, and **Secret** fields using the values that you saved during the *Configuring prerequisites* exercises.

Parameter	Value	
URI	https://graph.microsoft.com/beta/reports/credentialUserRegistrationDetails	
Queries	Key	Value
	\$filter	isMfaRegistered eq false
Authentication	Active Directory OAuth	
Tenant	<tenant ID value>	
Audience	https://graph.microsoft.com	
Client ID	<Application (client ID) value>	
Secret	<Client secret value>	

Table 15.1: Configuring HTTP action parameters

- 10. Review the completed action, as shown in *Figure 15.15*:

Generate Non-MFA Report

Method: GET

URI: https://graph.microsoft.com/beta/reports/credentialUserRegistrationDetails

Headers:

Enter key	Enter value
-----------	-------------

Queries:

Enter key	Enter value
\$filter	isMfaRegistered eq false

Body: Enter request content

Cookie: Enter HTTP cookie

Authentication: Active Directory OAuth

Authority: Default: https://login.windows.net/

Tenant: 5304a95c-5ee5-4443- [redacted]

Audience: https://graph.microsoft.com

Client ID: 749ca679-0085-4a55- [redacted]

Credential Type: Secret

Secret: h9XZQ~Q3\_4uGVuzn [redacted]

Figure 15.15: Reviewing the completed HTTP action

- 11. Click the **ellipsis** at the top of the HTTP action and select **Settings**:

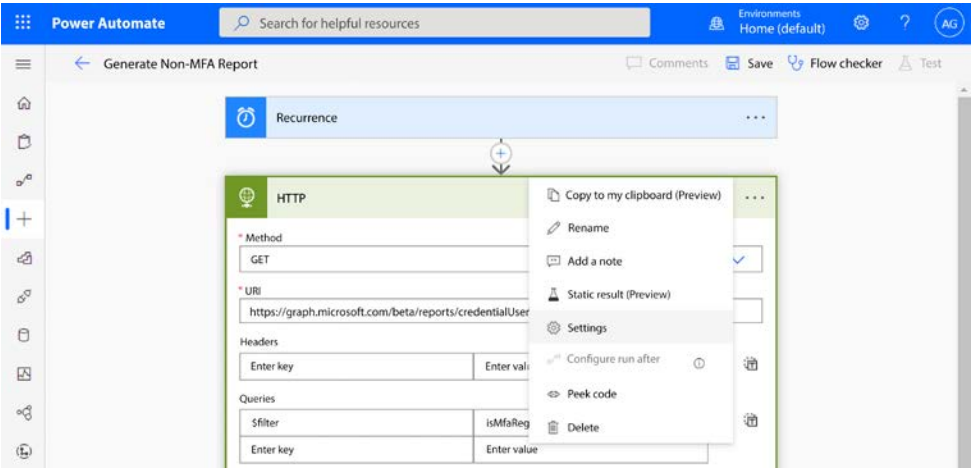


Figure 15.16: Configuring the HTTP action

- 12. Under **Pagination**, set the Pagination toggle switch to **On** and configure it with a **Threshold** value of 5000. Ensure the **Asynchronous Pattern** toggle switch is set to **On**:

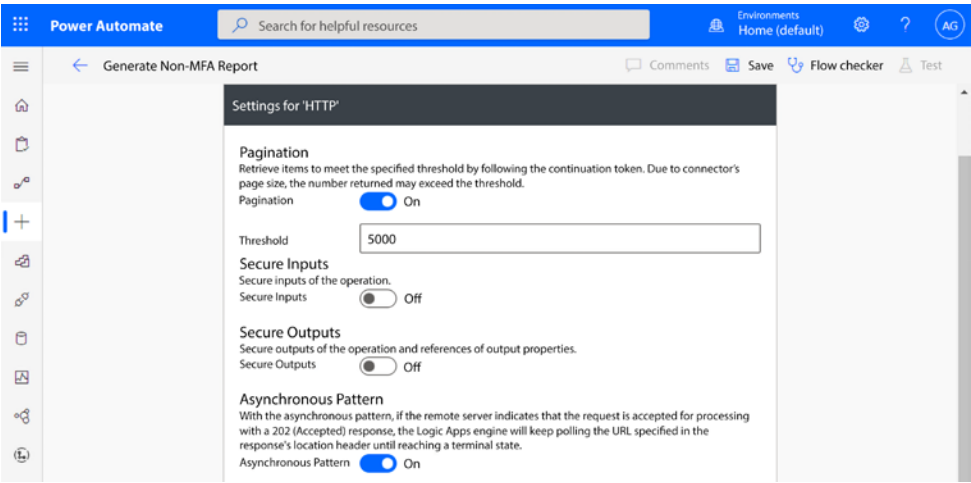
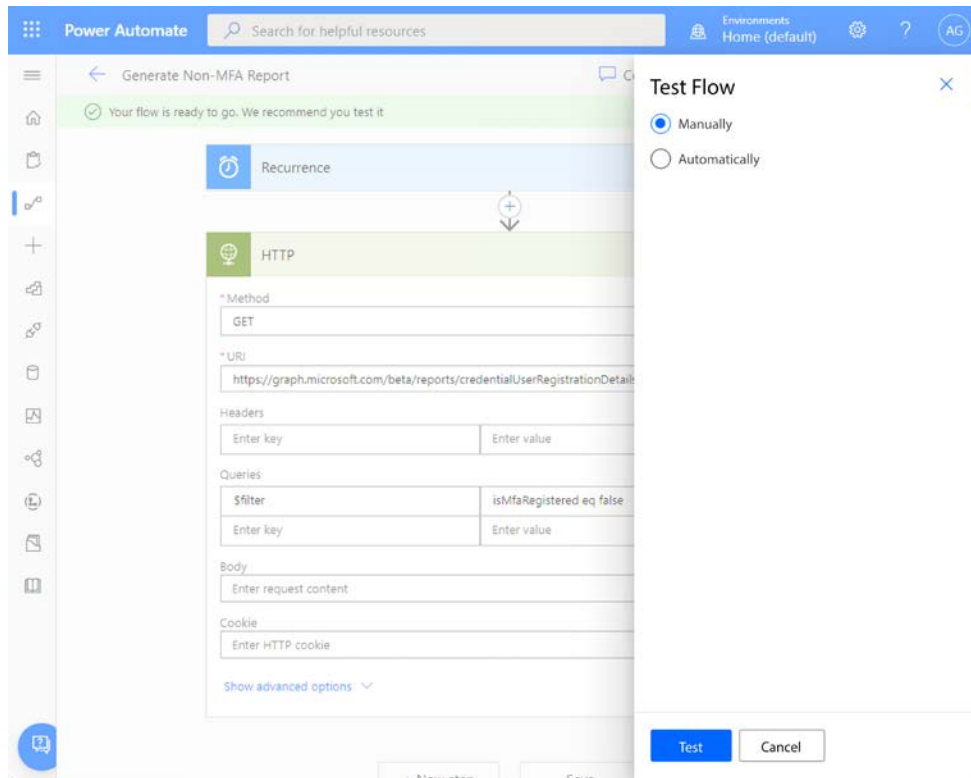


Figure 15.17: Configuring the HTTP action settings

- 13. Click **Done**.
- 14. Click **Save**.

15. Click **Test**, and then on the **Test Flow** flyout panel, select the **Manually** radio button and then click **Test**, as shown in *Figure 15.18*:



*Figure 15.18: Executing a quick test*

16. Confirm your choice by clicking **Run flow**.
17. After the flow runs, expand the **HTTP** action step by clicking on the step's title bar.
18. Scroll down to the **Outputs** section of the run history. Copy the entire **Body** value in the text box to the clipboard. This data will be used to define the JSON schema (organizational structure):



You may want to paste the data to a text document to verify that you captured all of the data, including the opening and closing braces.

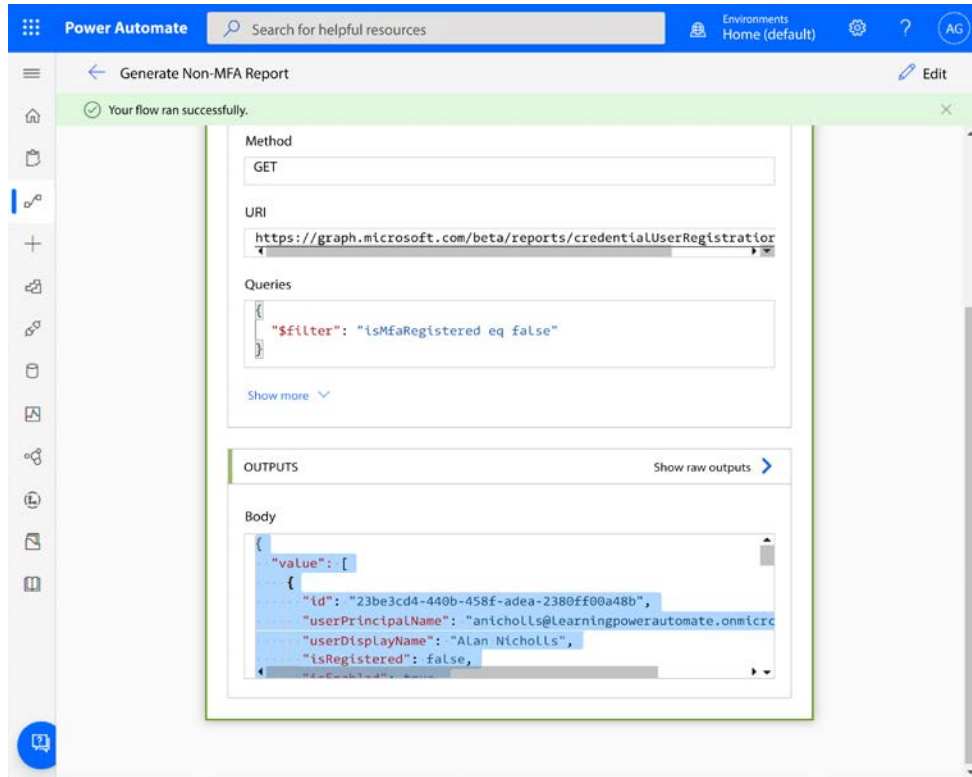


Figure 15.19: Copying the body value

19. Click **Edit** to return to editing the flow.

At this point, you've created a flow that successfully connects to Azure AD. Next, we'll look at configuring Power Automate with a Parse JSON action.

## Parsing the output

After successfully querying Azure AD for a list of users without MFA configured, you now need to configure a step to decode the data returned. This process, known as parsing, will allow Power Automate to identify individual records, as well as individual properties or parameters inside those records.

Follow these steps:

1. Scroll to the bottom of the current flow and click **New step**.
2. Select the **Parse JSON** data operation action:

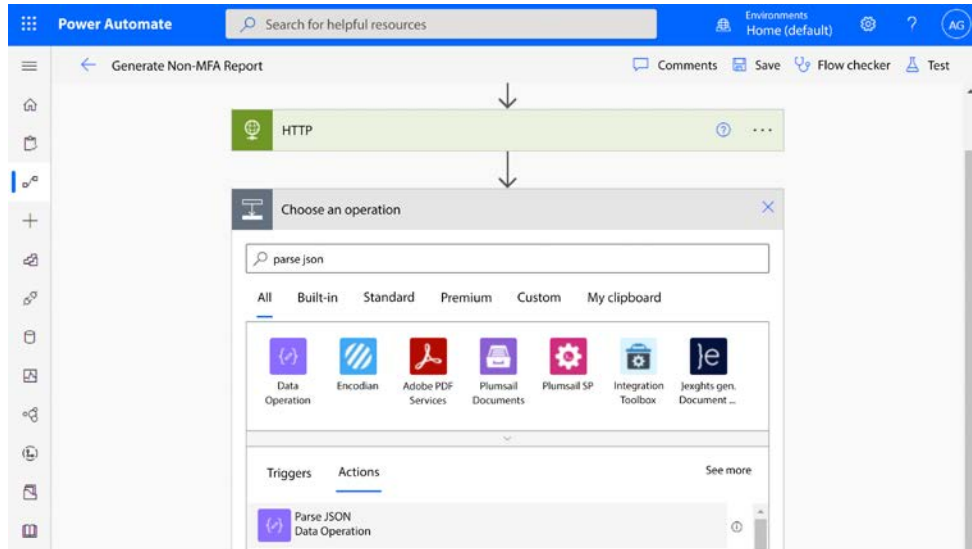


Figure 15.20: Adding the Parse JSON action

3. Select the **Content** field, and then add the **Body** dynamic content value, as shown in Figure 15.21:

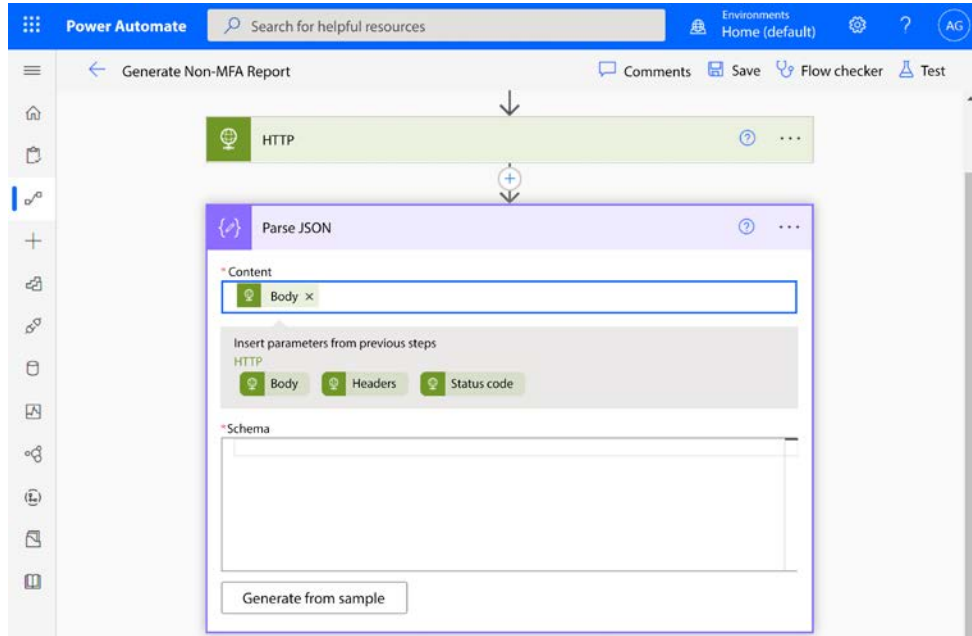


Figure 15.21: Configuring the Parse JSON action

4. At the bottom of the **Parse JSON** action, click **Generate from sample**.
5. Paste the JSON data copied during *step 18* of the previous section into the body of the **Insert a sample JSON payload** dialog box. Click **Done**:

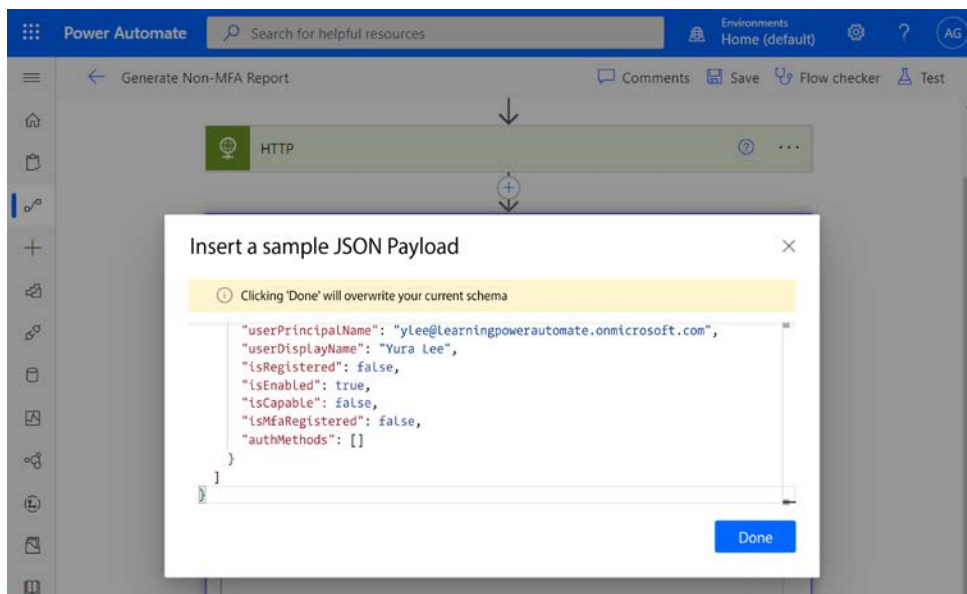
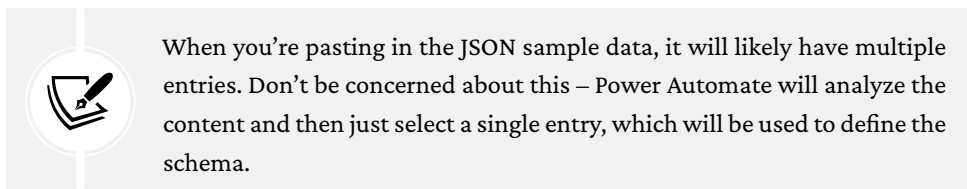


Figure 15.22: Using the JSON data to populate the schema

6. Review the schema data to ensure it has opening and closing braces. Click **Save**.

Power Automate now has the information required to interpret the output retrieved from Azure AD. Next, we'll take this output data and build it into a report that can be sent to the security team.

## Creating a report

The final portion of this process will be processing the JSON data and turning it into a report that the security team can read.

To complete the flow, follow these steps:

1. With the flow open, scroll to the bottom and click **New step**.
2. Select the **Filter array** data operation action.
3. In the **From** field, select the **value** dynamic content token in the **Parse JSON** section, as shown in *Figure 15.23*:

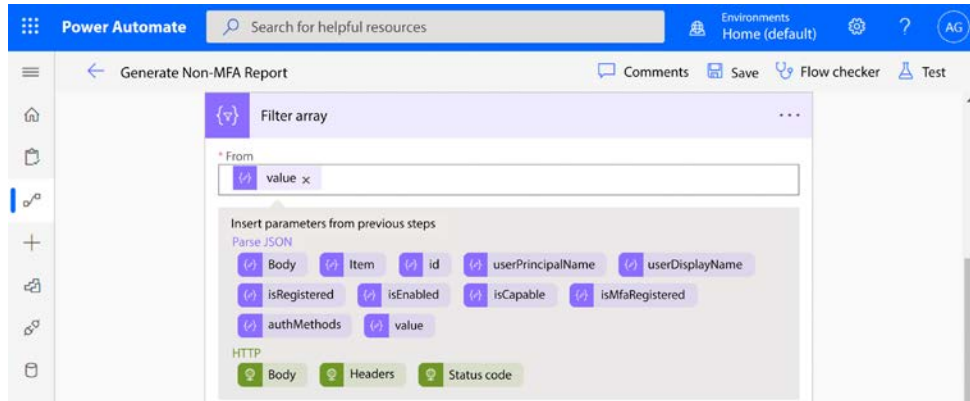


Figure 15.23: Configuring the Filter array

4. In the first **Choose a value** box, select the **Body** dynamic content value in the **Parse JSON** section.
5. In the **Operator** drop-down selection, choose **is not equal to**:

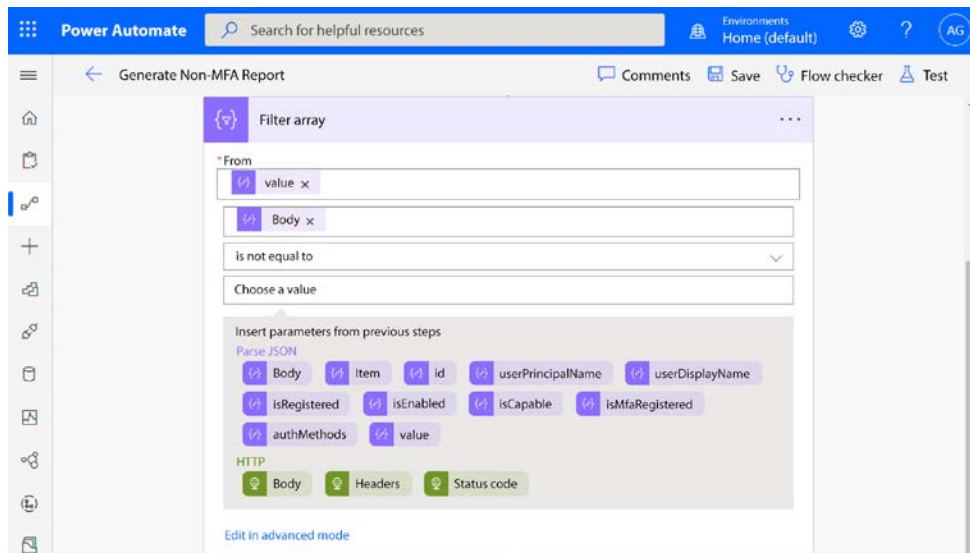


Figure 15.24: Configuring the Filter array action



6. In the second **Choose a value** box, select **Expression**, and then select **null**:

If you don't see the selection box that allows you to choose **Dynamic content** or **Expressions**, try changing your browser window size or browser zoom setting. If you can't get it to display, you can also click **Edit in advanced mode** and paste in the following text:

```
@not(equals(body('Parse_JSON'), null))
```

You'll need to ensure that it's pasted *exactly* like that in the box, paying special attention that the word `null` does not have any quote marks around it. Incorrect formatting will cause the word *null* to be treated as a string value instead of an expression. See *Figure 15.25* for a correct example.

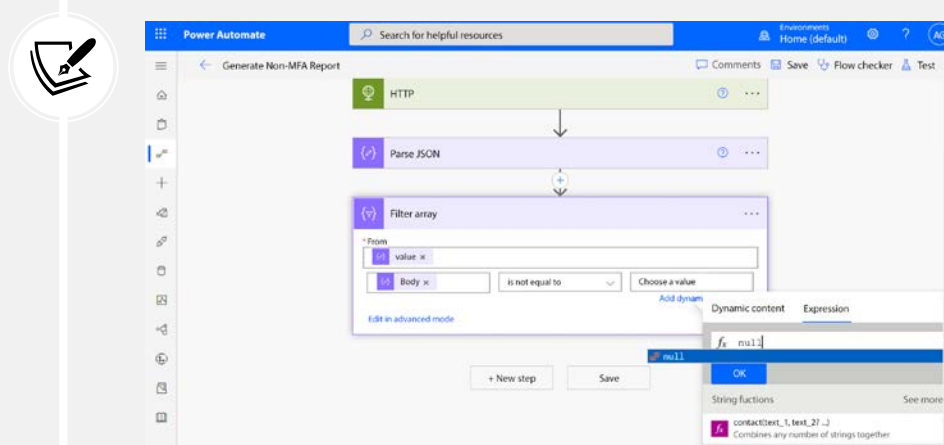


Figure 15.25: Adding the null expression

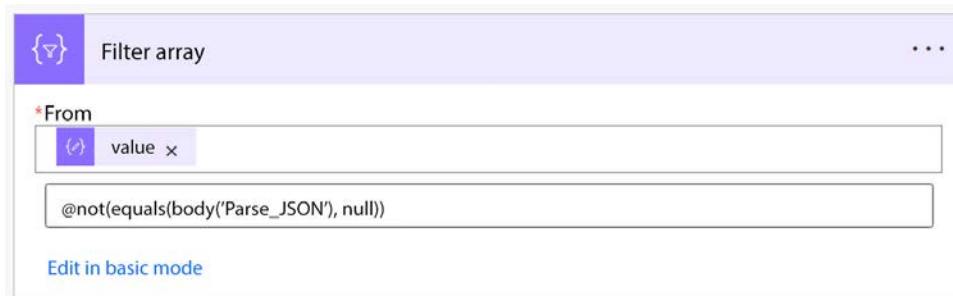


Figure 15.26: Entering the expression in advanced editing mode

7. Click **New step**.
8. Select the **Create CSV table** data operation:

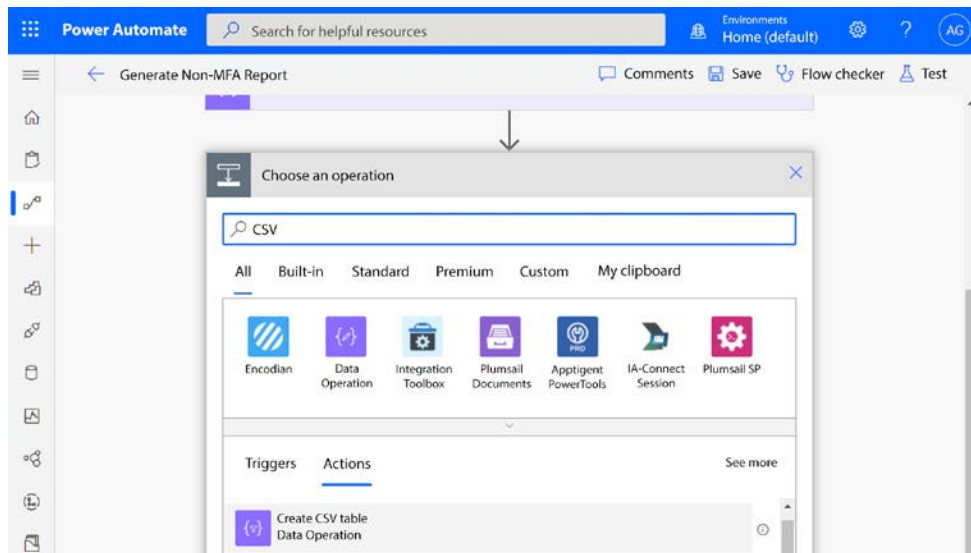


Figure 15.27: Adding the Create CSV table action

9. In the **From** field, select the **Body** dynamic content token from the **Filter array** section:

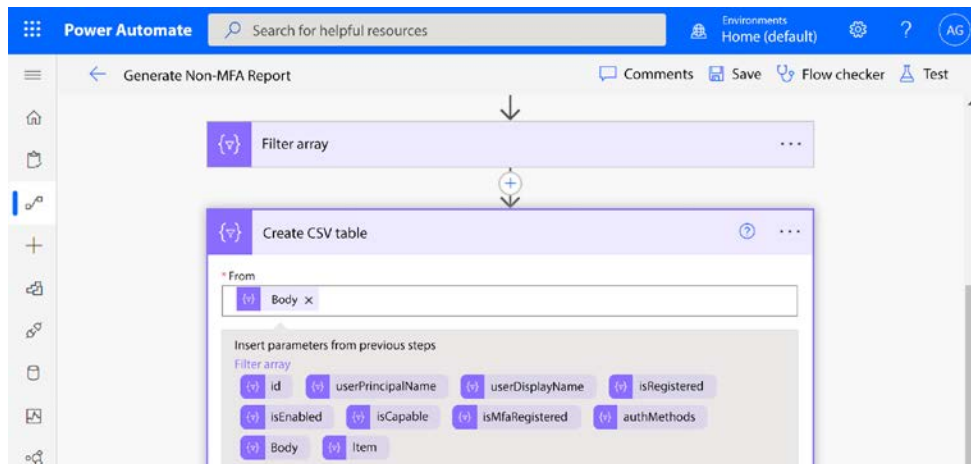


Figure 15.28: Configuring the Create CSV table action

10. In the **Create CSV table** action, select **Show advanced options**.
11. In the **Columns** drop-down, select **Custom**. We'll use this setting to define the columns to be added to the CSV table:

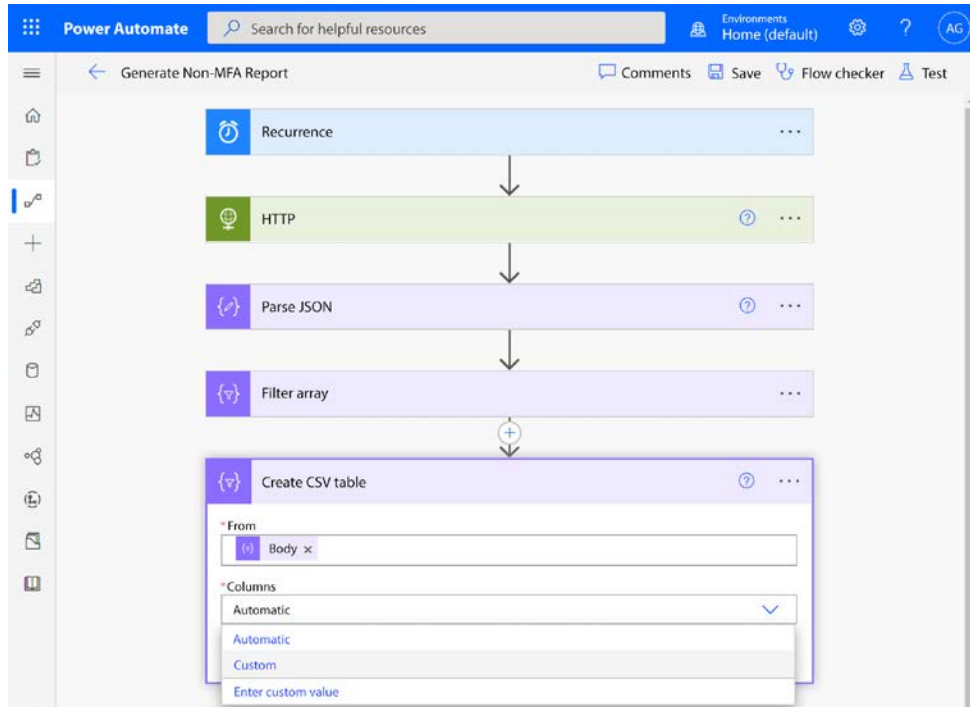


Figure 15.29: Enabling custom columns

12. Populate the **Header** column with the information you want to include and the **Value** column with the corresponding dynamic value token from the **Filter array** section. This example features the **DisplayName** header mapped to the **userDisplayName** token and the **Address** mapped to the **userPrincipalName** token:

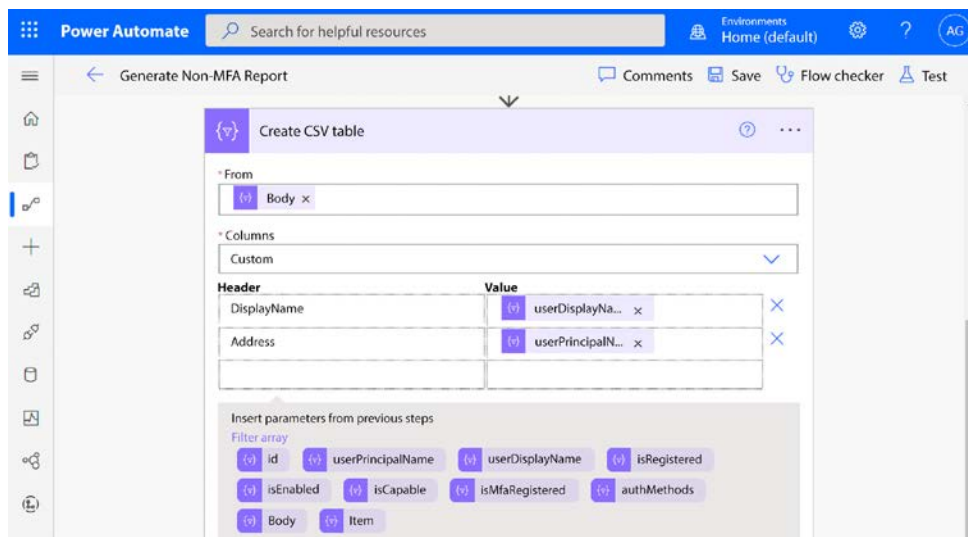


Figure 15.30: Configuring custom columns

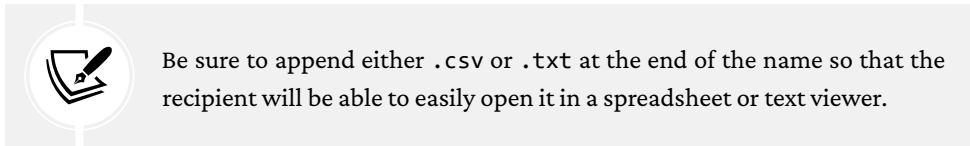
13. Click **New step**.
14. Select the **Send an email (V2)** action.
15. In the **To** box, enter an email address or select a recipient from the address book.



If you do not have an Office 365 Outlook connection configured, you will be prompted to enter credentials for a mailbox-enabled account before being able to continue.

16. Enter a **Subject**, such as Weekly report of users who have not completed MFA registration.
17. In the **Body** section, enter descriptive text, such as Weekly report of users who have not completed MFA registration.

18. Click **Show advanced options**.
19. In the **Attachments Name** box, enter a file name to give to the CSV output.



20. In the **Attachments Content** box, select the **Output** dynamic content token in the **Create CSV table** section, as shown in *Figure 15.31*:

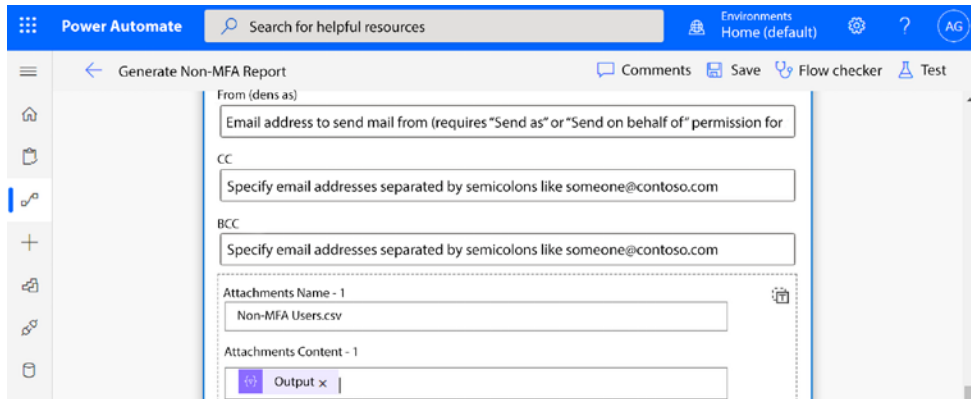


Figure 15.31: Configuring the attachment name and content

21. Click **Save**.

Congratulations! You've now configured an automated Azure AD report for users without MFA registration.

## Testing the flow

To test this flow, you can click **Test** in the upper-right hand corner of the Power Automate canvas area, as shown in *Figure 15.32*:

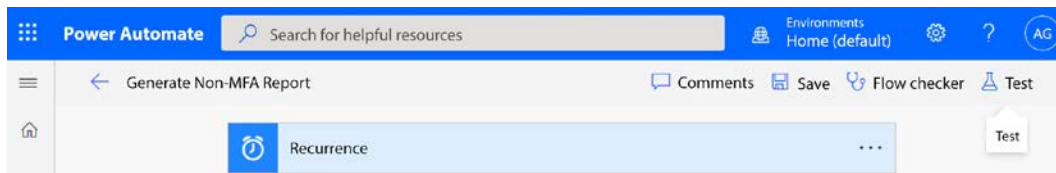
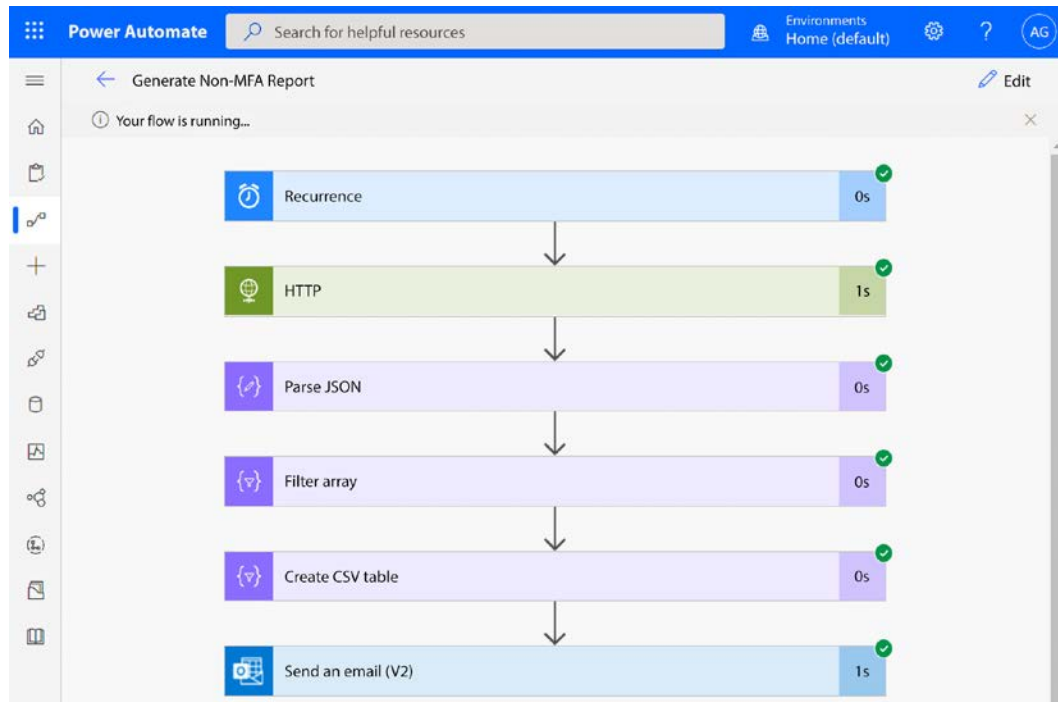


Figure 15.32: Testing the flow

Select the options to manually kick off a test flow. After successfully launching a test, you can watch the active run history in the Power Automate canvas area, as shown in *Figure 15.33*:



*Figure 15.33: Watching a flow run*

If any of the actions or steps return errors, you can expand the step and review the output.

# Verifying the flow

The final step is going to be reviewing the email that was sent. To do so, log into the mailbox that was used as the recipient for the **Send an email (V2)** action and look for the message:

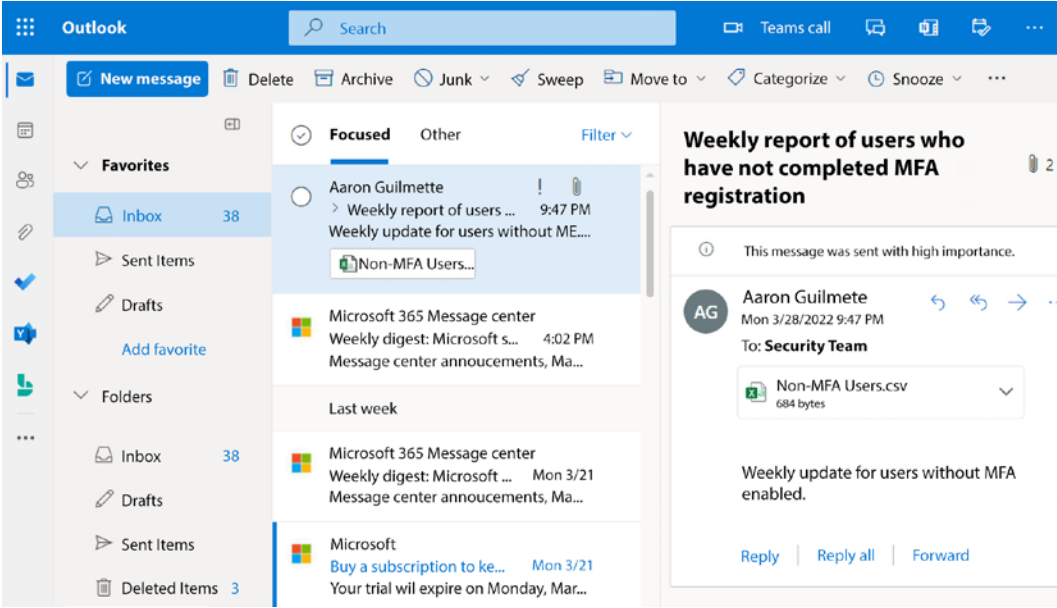


Figure 15.34: Reviewing the email message with the attached report

You can open the attachment and view the details, as shown in Figure 15.35:

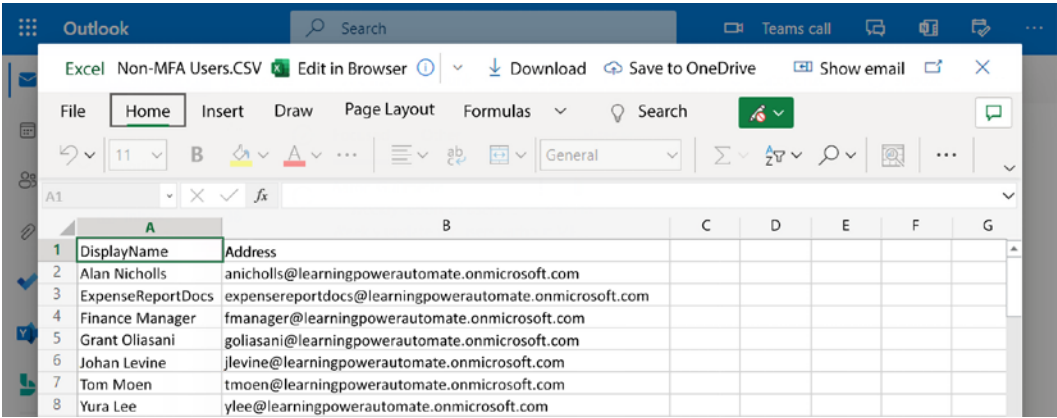


Figure 15.35: Examining the attachment

If everything worked correctly, you should have a report that includes all users in your Microsoft 365 tenant that do not have MFA enabled.

## Expanding further

In addition to just creating CSV output, you can also explore richer data alternatives. One such option is to create an HTML table to include directly in the email.

The **Create HTML table** data action functions identically to the **Create CSV table** action, with the exception that the output is basic HTML-formatted data instead of comma-separated data. You can potentially improve your email's usability by including the HTML output in the body of the email message.

## Summary

Being able to work with Azure AD opens up a lot of administration and reporting tasks to automation. Using Power Automate to help process reporting data or other administrative tasks can potentially free up precious time to work on other high-value tasks.

In this chapter, you learned how to start working with Azure AD directly. You learned how to interact with the Microsoft Graph API to retrieve data about MFA information, parse and filter it, and then add the output as a CSV attachment to an email. You were introduced to several new actions (including **HTTP**, **Parse JSON**, and **Create CSV table**) and used them to retrieve and format data in a way that could be easily interpreted by a person.

In the next chapter, we're going to start working with robotic process automation.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>







# 16

## Introducing Robotic Process Automation

As you've seen so far, Power Automate is an extraordinarily capable cloud automation platform. While Power Automate's focus had originally been to integrate cloud services through the use of REST API interfaces, Microsoft recognized a gap and released its tooling for **robotic process automation (RPA)** flows in mid-2020.

Originally named **user interface (UI)** flows, RPA flows are designed to capture and replay operations performed by manual intervention, such as data entry via keyboard and mouse clicks. RPA flows can help automate tasks where no scripted or API interface opportunities exist. Common examples might include:

- Duplicating data entry into multiple systems through copy and paste
- Reading handwritten order forms and translating them to an application
- Performing a series of keyboard presses or acknowledgments for legacy machinery

Each of these scenarios presents different types of challenges for a traditional scripting solution or REST-based interface to work with. RPA helps solve some of these problems by allowing you to essentially record actions and play them back. You can think of RPA as a macro that allows you to automate desktop applications and browser sessions.

RPA in Power Automate supports two types of automation methods: flows created with the Power Automate desktop application and flows built with **Selenium IDE**. Selenium IDE is an open-source tool that can be used to record Microsoft Edge and Google Chrome browser sessions. Since we're going to be working directly with a desktop application, we're going to use the Power Automate desktop application (which also has browser recording capabilities).

In addition, Power Automate supports two RPA automation scenarios: **attended** and **unattended**. Attended RPA scenarios include human interaction, either to initiate a flow or provide input during the execution. Unattended flows are fully automated processes that can run without user intervention.

In this chapter, we're going to explore working with RPA to take input from a Microsoft Forms survey and process it unattended. In *Chapter 13, Working with Microsoft Forms*, you took data and inserted it into a SQL database. In this chapter, we're going to take that Forms data and then enter it into a Microsoft Access database using a desktop flow. More specifically, we're going to tackle the following tasks:

- Learning about actions and variable passing
- Configuring prerequisites
- Configuring an RPA flow

By the end of this chapter, you should walk away with an understanding of some of the capabilities of the Power Automate desktop environment.

Let's go!

## Technical requirements

In addition to Power Automate licensing, this chapter will use the following software and applications:

- Power Automate Desktop (also called Power Automate for desktop)
- Microsoft Access
- The Microsoft on-premises data gateway

In this chapter, we'll be using the edition of Power Automate Desktop that ships with Windows 11. Power Automate Desktop is also available as a free download for Windows 10, which we'll cover in the *Configuring prerequisites* section.

Microsoft Access is available as a standalone app or as part of a suite, such as Microsoft 365 Apps for Business (formerly known as Microsoft Office 365 ProPlus) or Microsoft Office 2019. If you have a Microsoft 365 trial tenant, you can download an evaluation version of Microsoft 365 Apps from the landing page.

The Microsoft on-premises data gateway, like Power Automate Desktop, is available at no charge from inside the Power Automate web portal. We'll cover it as well in the *Configuring prerequisites* section later.



As a reminder, you'll need a Power Automate per user plan and the Unattended RPA add-on.

## Learning about actions and variable passing

In this chapter, we're going to introduce a few new concepts that relate to working with Power Automate Desktop, specifically around the action to run a Power Automate Desktop flow and how to pass variables from the cloud to a desktop application.

### Run a flow built with Power Automate for desktop action

The Power Automate Desktop application can't read or use cloud triggers. Using the connectivity established through the data gateway, however, Power Automate Desktop can receive the instructions it needs to execute the flows it's responsible for.

### Variable passing

**Variables** are a programming construct used to store temporary information. Programmers use variables as a kind of placeholder for content that changes (or varies) when an application, script, or process is run. Through the use of a variable, an application can substitute the actual value in place of the variable during execution.

You've already seen variables in use throughout this book, although the Power Automate user interface commonly refers to them as **dynamic content**. As you add actions in a Power Automate flow, the dynamic content tokens or variables are automatically built based on the output of the previous step.

Power Automate Desktop, however, has no way to automatically import those variables from a cloud flow. To help bridge that gap, Power Automate allows you to define your own variables to map that content between environments.

Power Automate has two kinds of variables: input and output. Input variables are new variables that you create based on the information exposed from the Power Automate service. Output variables are the resultant data that Power Automate Desktop can send back to the Power Automate service. When we begin configuring the flow, you'll learn how to create new variables to pass data between environments.

## The Power Automate recorder

The Power Automate recorder is a powerful tool that allows you to capture the steps you'd perform as a user, and it's a great way to get a feeling for some of the capabilities of the Power Automate Desktop environment.

If you've recorded a macro in Microsoft Excel before, you'll have an idea of how the Power Automate recording process works. Power Automate records behaviors such as launching applications, selecting particular dialog boxes or interfaces, entering text, and more. After the recording process completes, you're left with a list of actions that you can edit to refine your final output.

Now that you have an idea of what we'll be using, let's move on to configuring the prerequisites.

## Configuring prerequisites

As mentioned earlier, this flow will require a few software prerequisites – namely, the Power Automate Desktop application and the data gateway application. It will also require an Access database.



While it is possible to complete these steps on your main computer, we recommend using a second computer (or virtual machine) for configuring, prerequisites, and running the flows. Since Power Automate Desktop will automate mouse and keyboard inputs, you may experience difficulties and unexpected errors if you try to trigger the flow from the same computer that will be executing the actions.

If you don't have access to a second computer, you can start a no-charge Azure trial subscription (<https://portal.azure.com>) and provision a virtual machine. Alternatively, you can trigger the flow from another device you have access to, such as a phone or tablet.

Let's begin!

## Configuring Power Automate Desktop

Power Automate Desktop is already included with Windows 11, which is what we'll be using for the purposes of our exercises. However, you can download and install Power Automate Desktop on Windows 10 at no charge from <https://go.microsoft.com/fwlink/?linkid=2102613> or the Power Automate web portal, as shown in *Figure 16.1*:

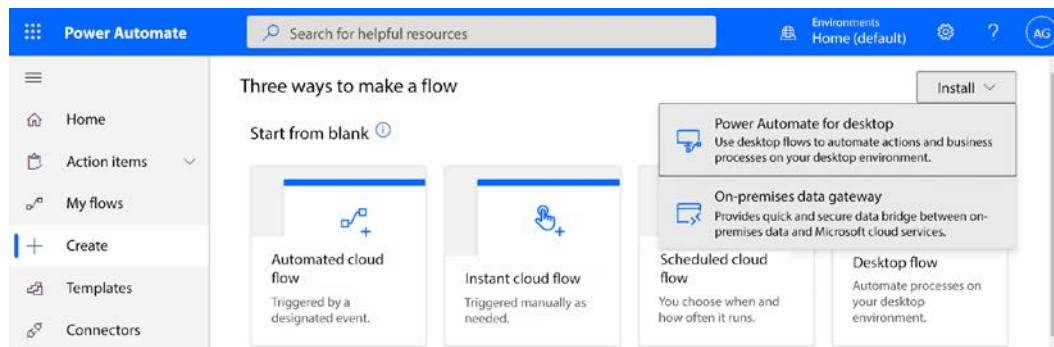
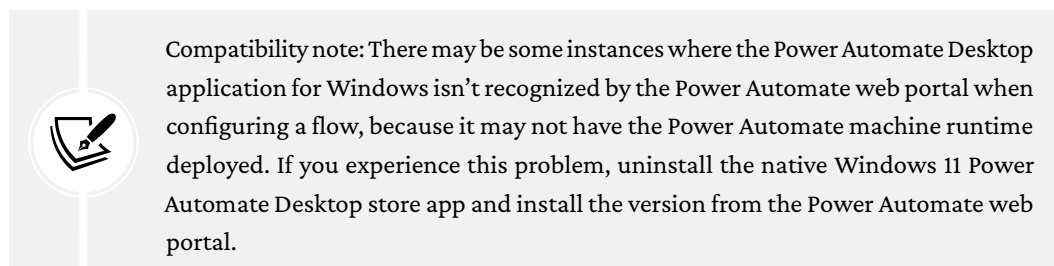


Figure 16.1: Downloading the Power Automate Desktop application



## Configuring a Microsoft Access database

In this chapter, we're going to work with the default contact database template that is included with Microsoft Access. To configure the database, follow these steps:

1. Launch Microsoft Access.
2. On the **Home** tab, under **New**, select **Contacts**:

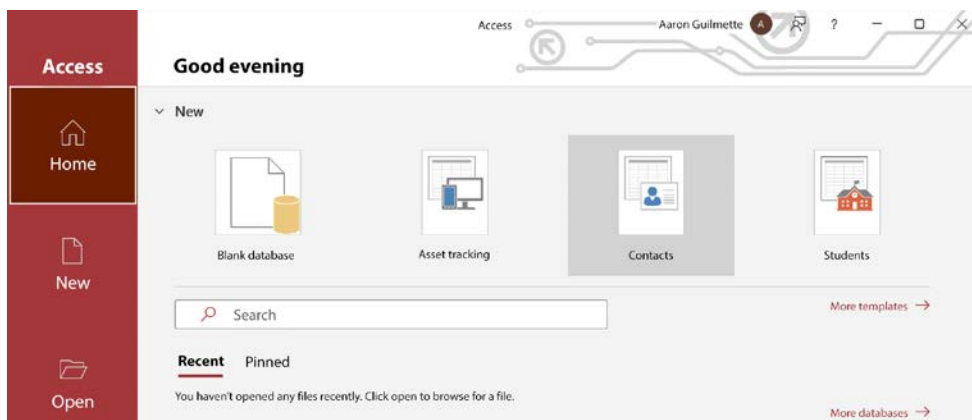
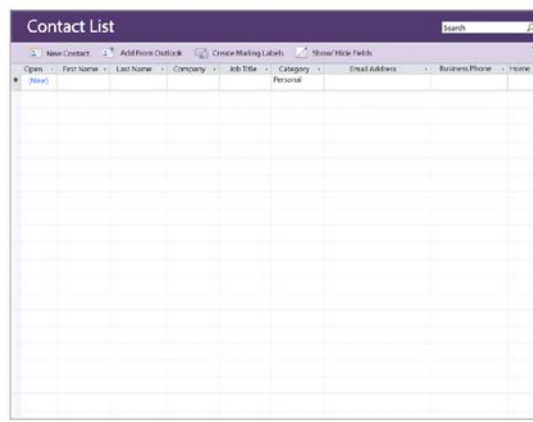


Figure 16.2: Creating a contact database

- 3. Accept the defaults and click **Create**:




### Contacts

Provided by: **Microsoft Corporation**

Create and maintain a comprehensive database of your customers partners and vendors using this popular access template. In addition to maintaining details on company, job title, and contact information, you can create all manner of queries, add contacts from Outlook, create mailing labels, and generate reports such as directories and phone books.

Download size: 143 KB

File Name  
 

C:\Users\labadmin\Documents\




Figure 16.3: Creating a contact database

- 4. Look around and familiarize yourself with the database. You'll notice that the database has a couple of key areas, including **Tables** (**Contacts** and **Settings**) and **Forms** (**Contact Details**, **Contact List**, and **Welcome**):

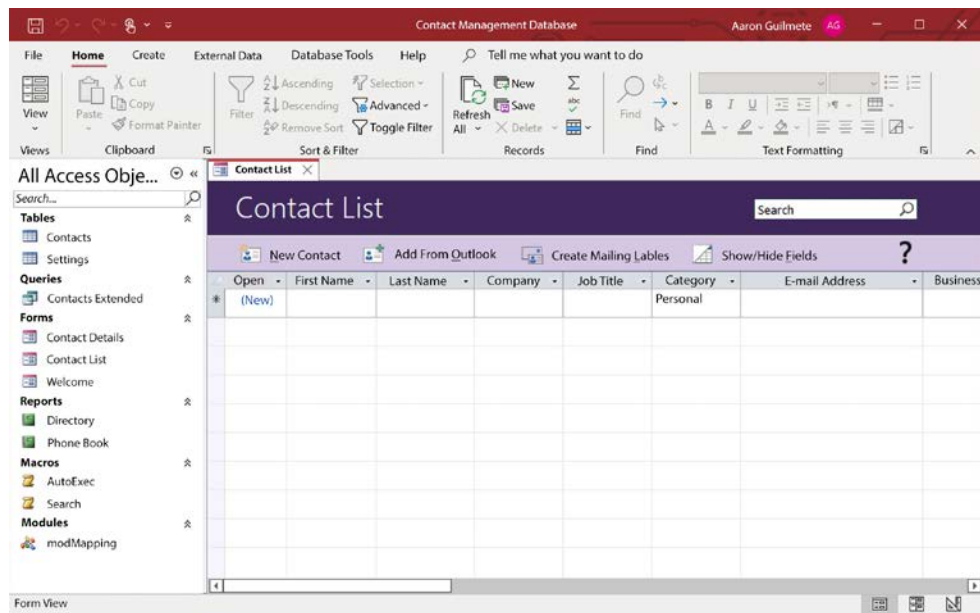


Figure 16.4: Viewing the Contact List form

5. We'll be working with the **Contact List** form in this chapter. Clicking the **New Contact** button on this page opens the **Contact Details** form shown in *Figure 16.5*, where we'll insert the Microsoft Forms data that we will capture later:

Figure 16.5 shows the 'Contact Details' form in Microsoft Outlook. The form is titled 'Untitled' and has a purple header bar. Below the header is a toolbar with buttons for 'Go to', 'Save and New', 'E-mail', 'Save As Outlook Contact', and 'Close'. The main content area is divided into two sections: 'General' and 'Notes'. The 'General' section contains fields for 'First Name', 'Last Name', 'Company', 'Job Title', 'E-mail', 'Web Page', 'Category', 'Business Phone', 'Home Phone', 'Mobile Phone', and 'Fax Number'. There is also a placeholder for a profile picture with an 'Edit Picture' button. The 'Notes' section has a 'Click to Map' button and a large text area for notes.

Figure 16.5: Viewing the Contact Details form



6. Close the **Contact Details** form. You'll want to close and reopen the database and clear the **Show Welcome when this database is opened** checkbox, as shown in *Figure 16.6*, as it will interfere with the automation:

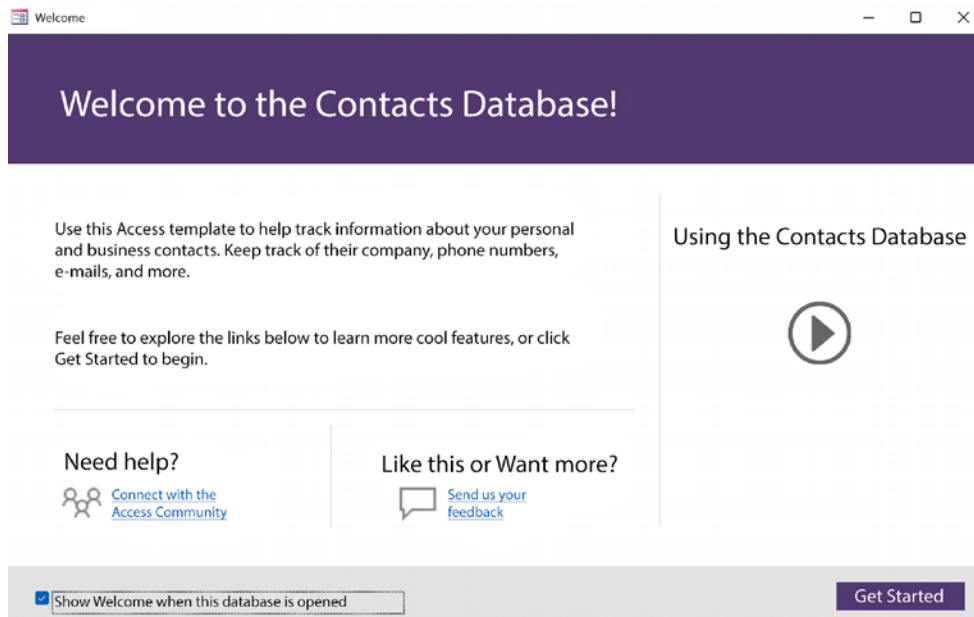


Figure 16.6: Welcome page for Contacts Database

7. When you're finished, exit the Microsoft Access application.

Next, we'll work on configuring the data gateway.

## Configuring the Microsoft on-premises data gateway

The data gateway's function is to allow devices on your internal network to communicate with the Power Platform services, such as Power Apps and Power Automate.

If you are working as part of an established organizational Microsoft 365 environment, you may need permission to install and configure the data gateway on your device and in the Microsoft 365 tenant. If you are working with your own equipment and a test tenant, you can follow these steps to configure the data gateway:

1. On the device you wish to install the gateway on, navigate to the Power Automate web portal (<https://flow.microsoft.com>):

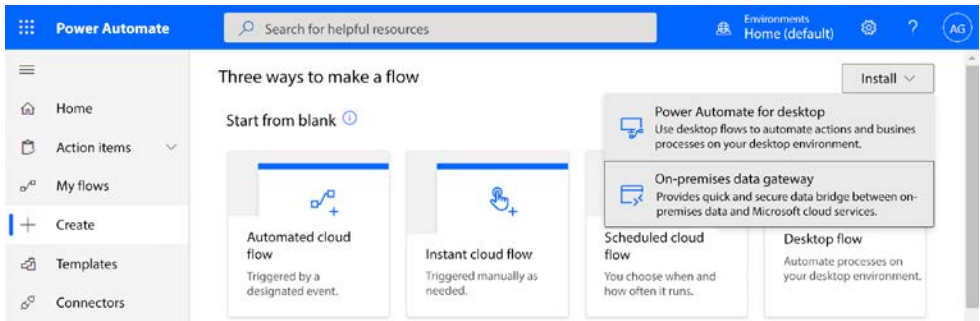


Figure 16.7: Downloading the On-premises data gateway app

2. Agree to the terms of use and then click **Install**:

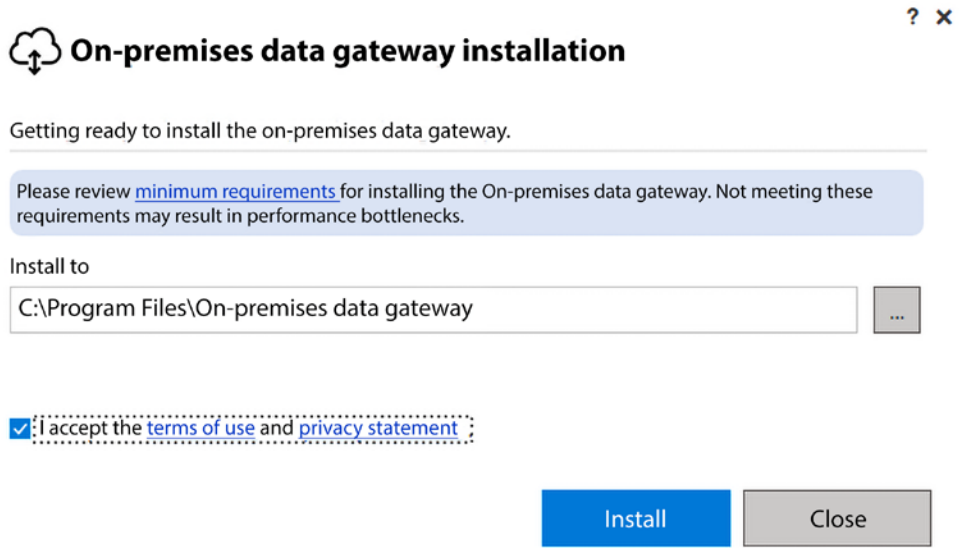
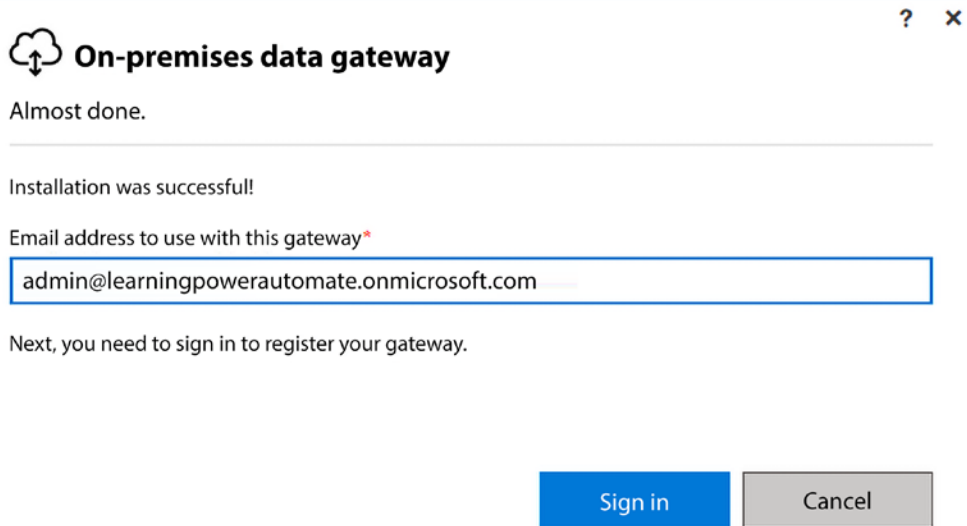


Figure 16.8: Installing the data gateway

3. Enter your Microsoft 365 credentials to begin registering the gateway with your tenant. Click **Sign in** to start the process:



**On-premises data gateway**

Almost done.

Installation was successful!

Email address to use with this gateway\*

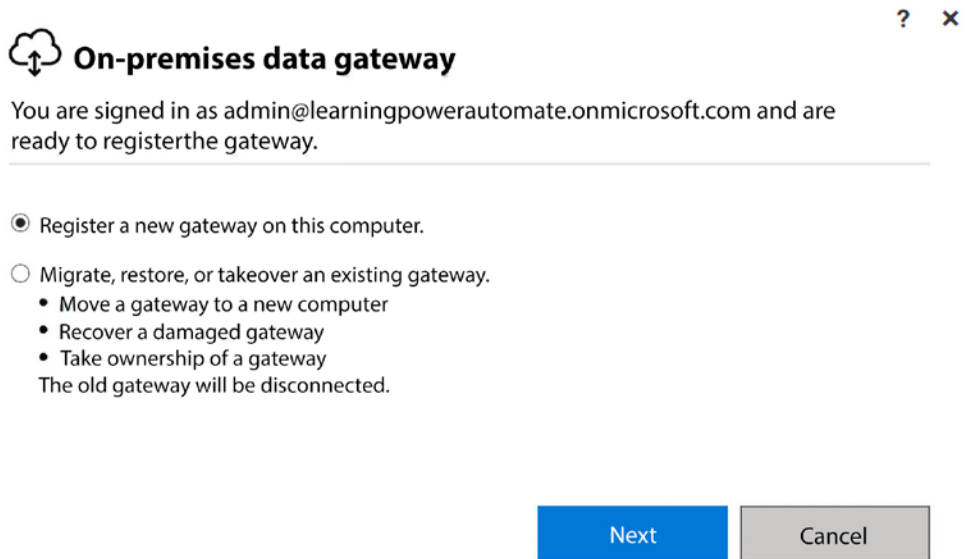
admin@learningpowerautomate.onmicrosoft.com

Next, you need to sign in to register your gateway.

Sign in Cancel

Figure 16.9: Configuring the data gateway

4. Select the option to **Register a new gateway on this computer** and click **Next**:



**On-premises data gateway**

You are signed in as admin@learningpowerautomate.onmicrosoft.com and are ready to register the gateway.

☒ Register a new gateway on this computer.

☐ Migrate, restore, or takeover an existing gateway.

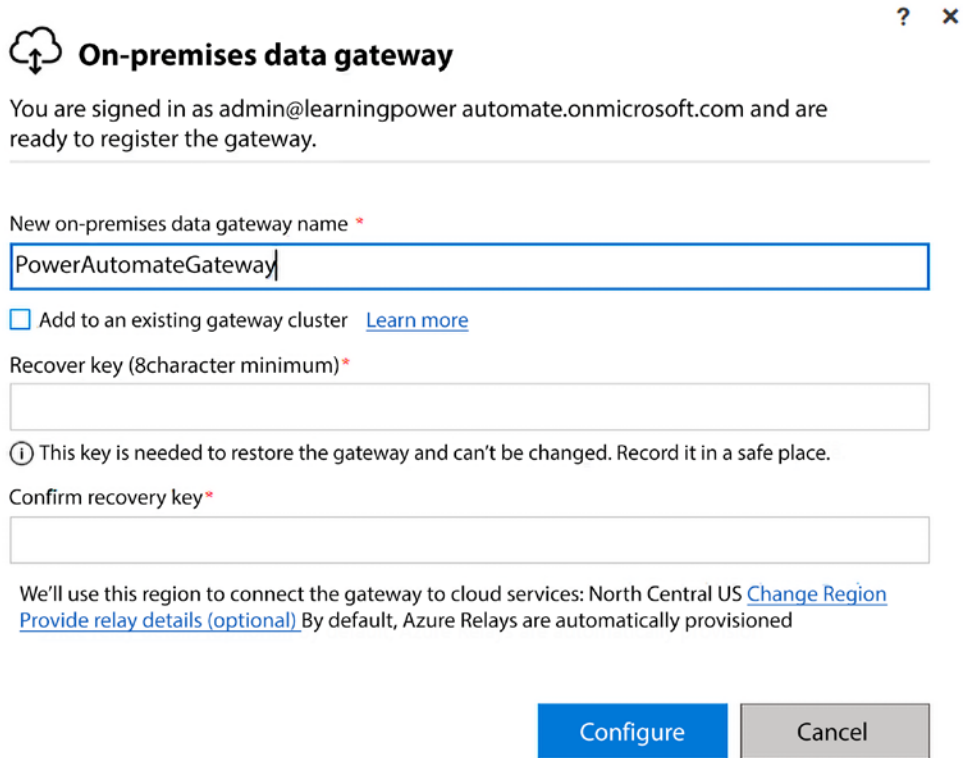
- Move a gateway to a new computer
- Recover a damaged gateway
- Take ownership of a gateway


The old gateway will be disconnected.

Next Cancel

Figure 16.10: Registering a new gateway

5. Enter a **New on-premises data gateway name**. This can be any name you like and doesn't have to be the name of the computer. This is the name that will be displayed when you configure a flow to access on-premises services in the Power Automate web portal:



 **On-premises data gateway** ? ×


You are signed in as admin@learningpower automate.onmicrosoft.com and are ready to register the gateway.

New on-premises data gateway name \*

PowerAutomateGateway

☐ Add to an existing gateway cluster [Learn more](#)

Recover key (8character minimum) \*

 This key is needed to restore the gateway and can't be changed. Record it in a safe place.

Confirm recovery key \*

We'll use this region to connect the gateway to cloud services: North Central US [Change Region](#)  
[Provide relay details \(optional\)](#) By default, Azure Relays are automatically provisioned

**Configure** **Cancel**

Figure 16.11: Configuring the gateway settings

6. Enter a value for **Recovery key**. The recovery key is required if you need to restore or reconfigure the gateway. Enter the same value in the **Confirm recovery key** box and click **Configure**.
7. After the installation is complete, click **Close**.

With the core prerequisites configured, let's move on to creating the flow!

## Configuring an RPA flow

For simplicity's sake, we're going to reuse the existing Microsoft Forms form that we created in *Chapter 13, Working with Microsoft Forms*. You can create a new one if desired, but you may need to modify the steps in the example to suit your form.

Since this is a more complex and involved flow, this process is going to be broken into a series of smaller sections.

## Configuring the flow framework

In this first section, we're going to configure a new flow to invoke an action launching Power Automate Desktop. We'll also set up the variables.

To get started, follow these steps:

1. On the computer where Microsoft Access and Power Automate desktop are configured, ensure the Contact Database is open with Microsoft Access.
2. Click **Create** and then select **Automated cloud flow**. Enter a **Flow name**, such as Add Contact to Access Database.
3. Select the **When a new response is submitted** Microsoft Forms trigger and click **Create**:

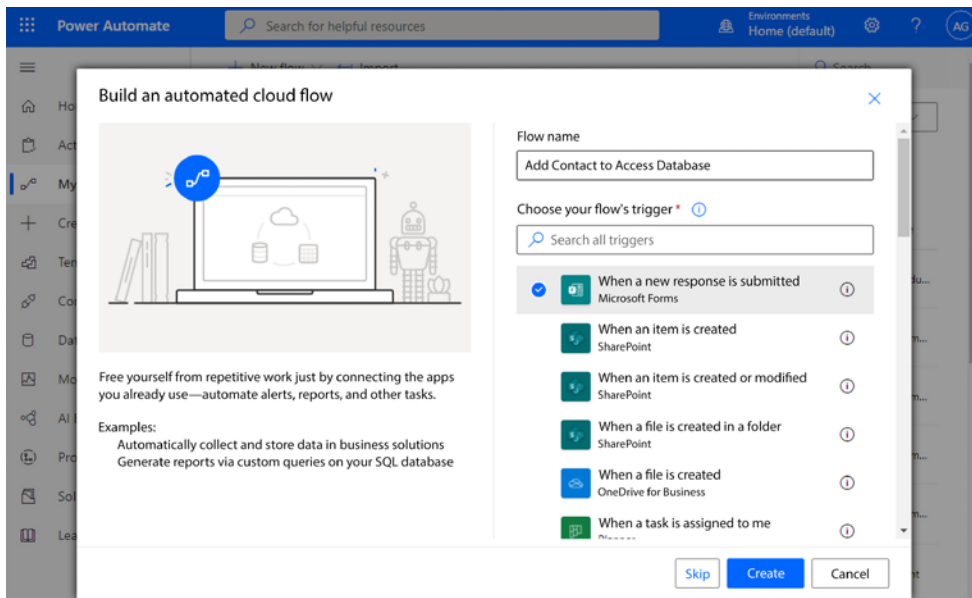


Figure 16.12: Adding a new flow

4. In the **When a new response is submitted** trigger, select the **Form Id** that corresponds to the **Customer Survey** form from *Chapter 13, Working with Microsoft Forms*.

5. Click **New step**, and then add the **Get response details** Microsoft Forms action.
6. Select the **Form Id** that you chose in the **When a new response is submitted** trigger. In the **Response Id** field, add the **Response Id** dynamic content value:

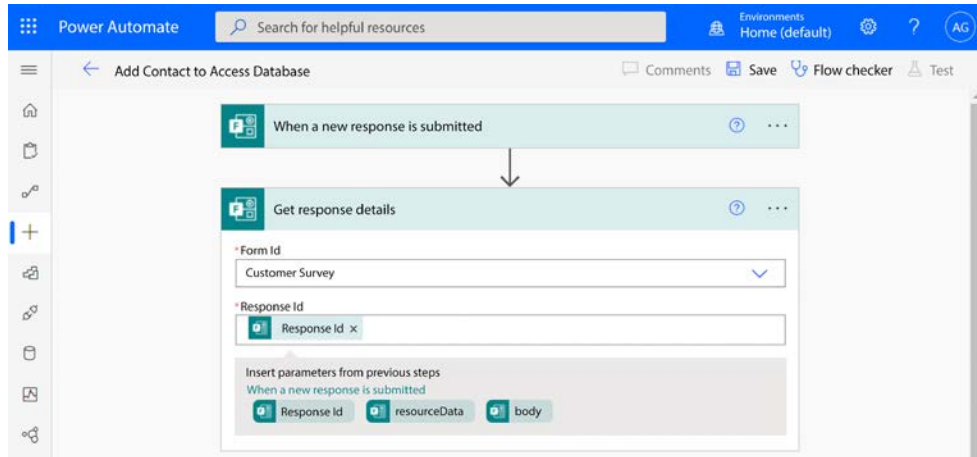


Figure 16.13: Adding the Get response details action

7. Click **New step**, and then select the **Run a flow built with Power Automate for desktop** action, as shown in Figure 16.14:

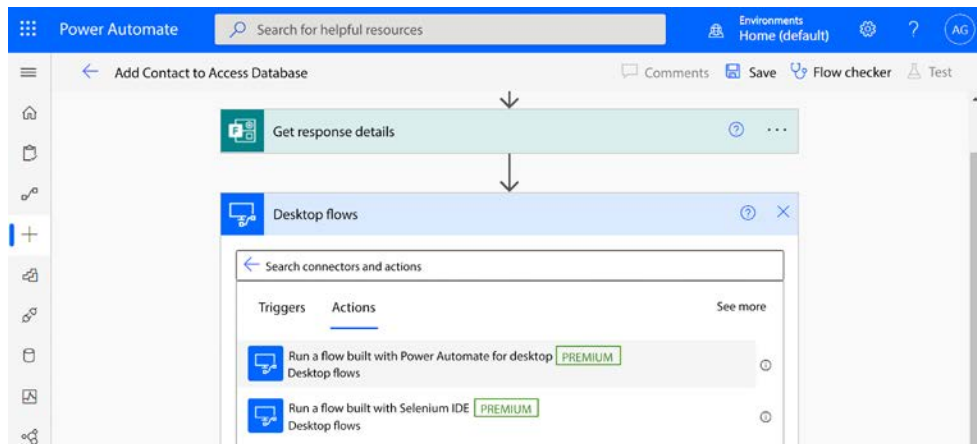


Figure 16.14: Adding the Desktop flows action

8. Populate the action's properties. Under **Connect**, select **Using an on-premises data gateway**. Under **Gateway name**, select the name of the gateway you configured in the *Configuring the Microsoft on-premises data gateway* section. Finally, enter the credentials you use to log onto the machine (either as DOMAIN\username or MACHINE\username. When finished, click **Create**. See *Figure 16.15* for a completed example:

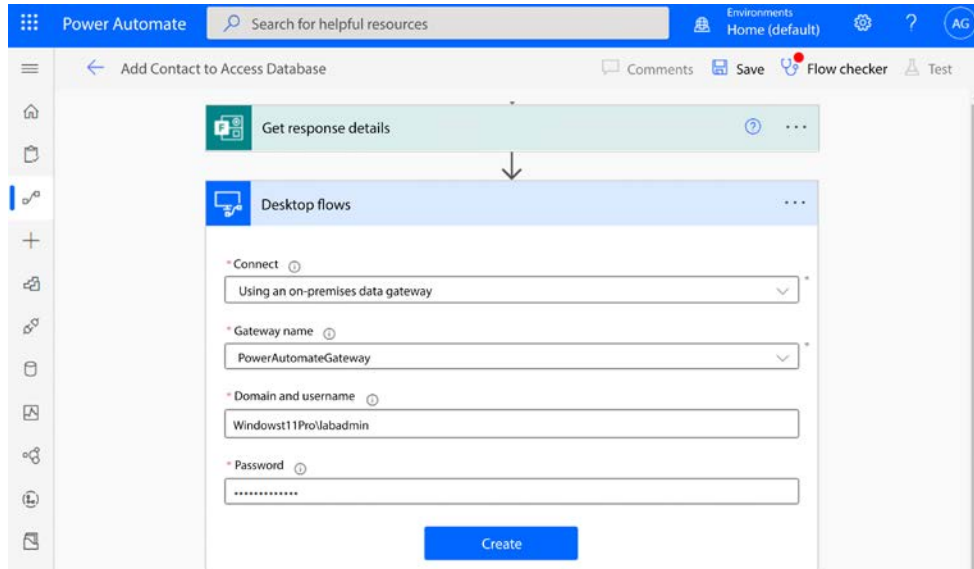


Figure 16.15: Configuring the Desktop flows action

9. After the connection object has been created, the action will update. Under **Desktop flow**, select the option to **Create a new desktop flow**:

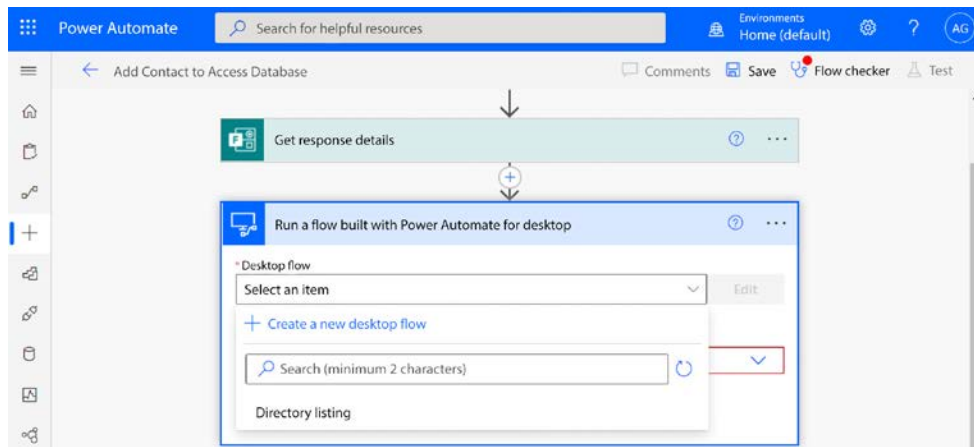


Figure 16.16: Configuring the Desktop flow action

10. Click **Launch app** to open the Power Automate Desktop app.
11. If prompted, click **Open** to allow the website to launch the local application.
12. If prompted, provide credentials to allow the Power Automate desktop app to connect to your environment. By default, you'll connect to the default environment for your Microsoft 365 tenant.
13. In the **Variables** pane, under **Input / output variables**, click the + and then select **Input** to add an input variable:

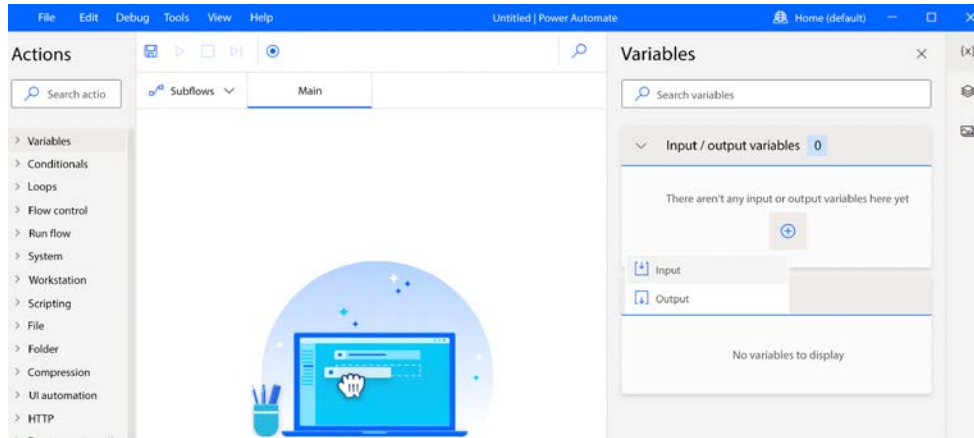



Figure 16.17: Adding variables

14. Fill out the input variable screen. See Figure 16.18 for a completed example:




**Variable name** and **External name** both refer to the name that you'll give a particular variable. The **Variable name** property is the name of the variable that you'll use when you're working in Power Automate Desktop. The **External name** property is what will show up in the Power Automate cloud service.

- **Variable type:** Input
- **Variable name:** varFormCompanyName
- **Data type:** String
- **Default value:** <empty>
- **External name:** varDesktopCompanyName
- **Description:** <empty>



- **Mark as sensitive:** <off>

Add a new input variable
×

 Add a new variable to be used as input or output [More info](#)

Variable type:

Input

i

Variable name:

varFormCompanyName

i

Data type:

String

i

Default value:

Add a default value

i

External name:

varDesktopCompanyName

i

Description:

Add an input description

i

Mark as sensitive

☐

i

Create

Cancel

Figure 16.18: Filling out the variable parameters

15. The Microsoft Forms form should have six data fields (CompanyName, FirstName, LastName, JobTitle, Mail, and TelephoneNumber). Repeat steps 13 and 14 for each of the remaining variables.
16. Click **Save** to save your progress.

Next, we'll use Power Automate to record the steps to input data.

## Configuring an application to open

In order to ensure success, it is recommended to start with a **System** action to run the Microsoft Access application. To configure the action, follow these steps:

1. With the Power Automate Desktop designer open, in the **Actions** pane, search for Run.
2. Select the **Run application** action and drag it to the **Main** body area of the flow:

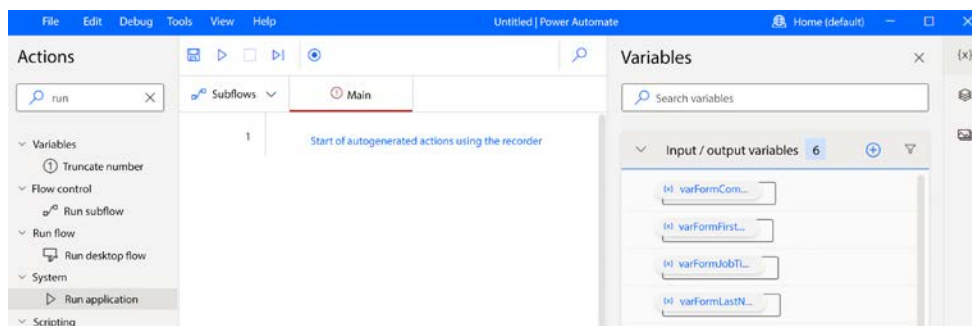


Figure 16.19: Adding a Run application action

3. In the **Application path** box, browse to the location of the Microsoft Access executable – for example, `C:\Program Files\Microsoft Office\root\Office16\MSACCESS.EXE`:

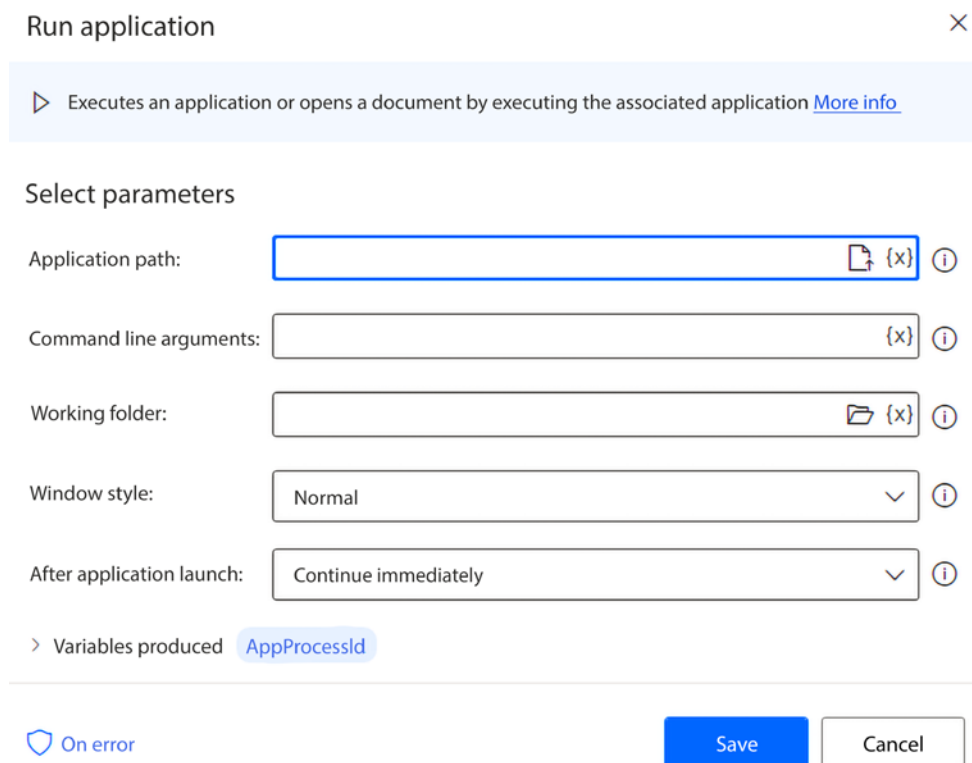


Figure 16.20: Adding the Microsoft Access application path

4. In the **Command line arguments** box, add the path of the contacts database file. You can use Windows Explorer to navigate to the folder containing the database, right-click the database file, and select **Copy as path**:

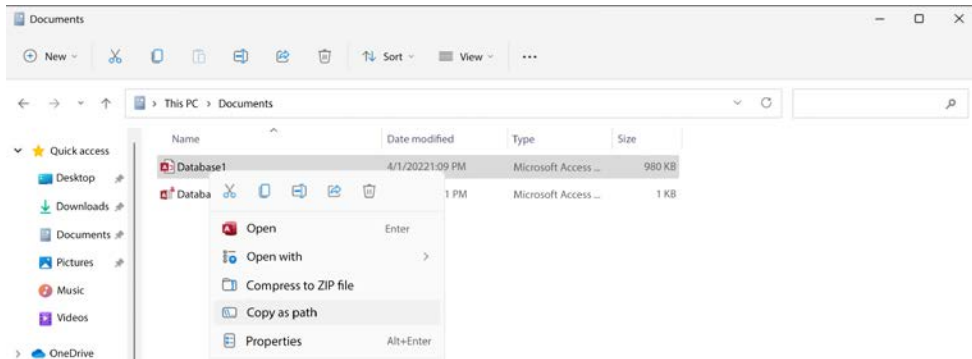


Figure 16.21: Copying the file path for the database

5. Paste that value in the **Command line arguments** box, as shown in Figure 16.22:

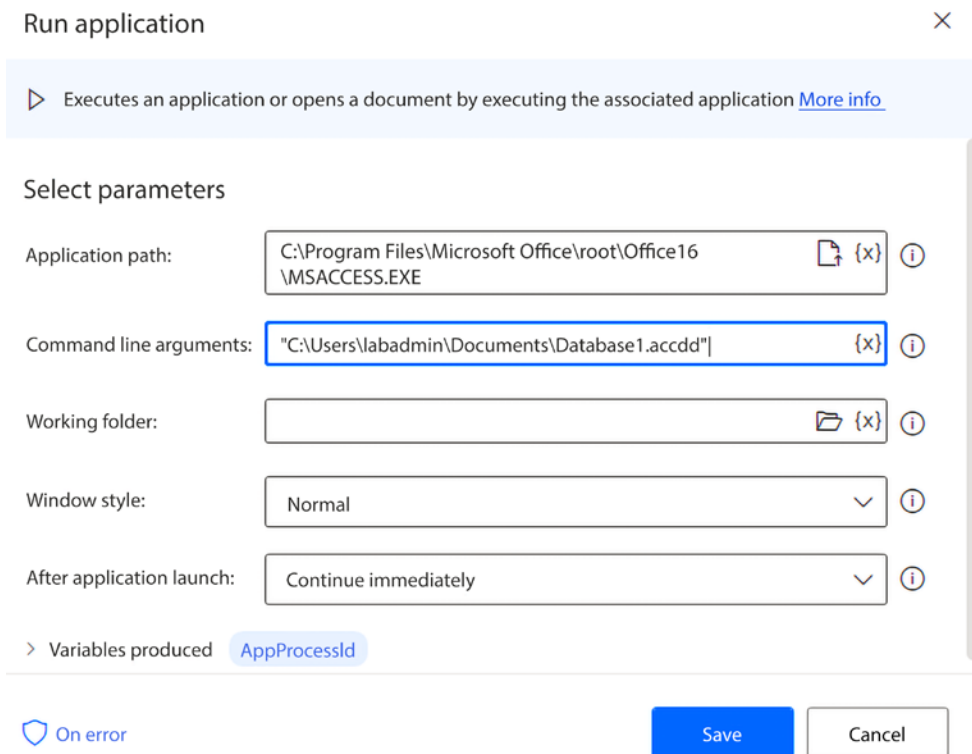


Figure 16.22: Adding the database path as an argument

6. Click **Save** to save the step.

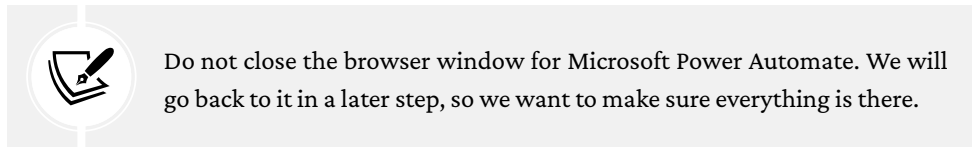
At this point, you have configured the desktop flow to launch Microsoft Access and open the correct database file.

## Running the Power Automate recorder

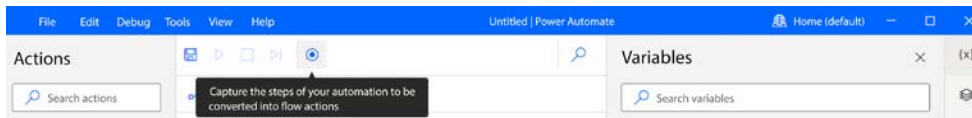
Now, we're ready to start capturing the basic actions that will make up the flow.

Let's get started!

1. Ensure no unnecessary desktop applications are visible on the desktop. You can either exit or minimize them so that they don't interfere with the recording process.



2. With the Power Automate Desktop app open, click the **Record** button on the menu bar:



*Figure 16.23: Starting the Power Automate recorder*

3. Click the **Record** button in the **Recorder** app. From this point on, the Power Automate recorder will capture every activity you perform on the desktop (with the exception of clicking inside the **Recorder** application or moving it):

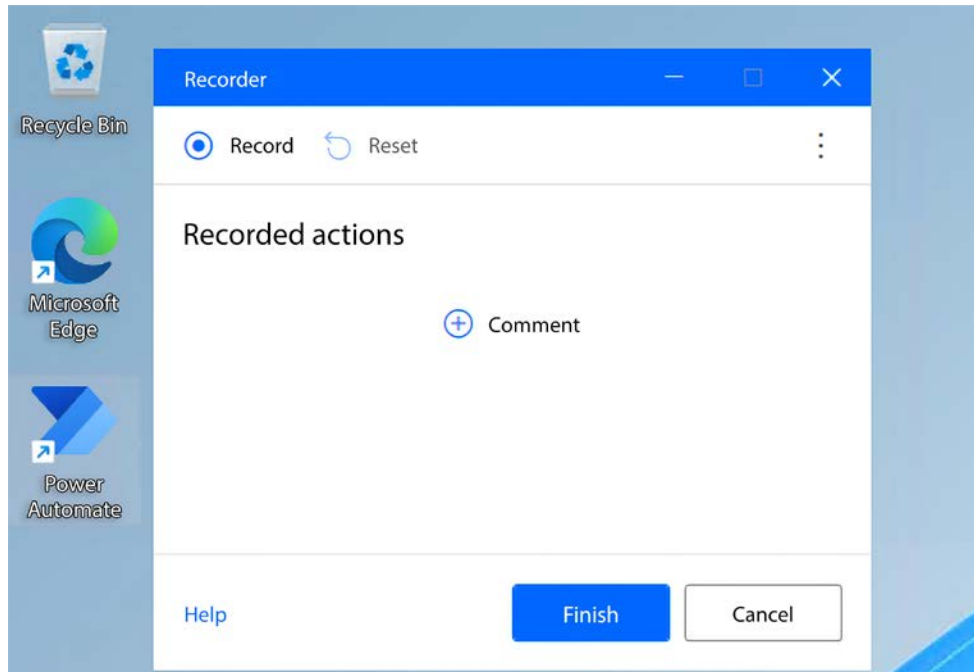
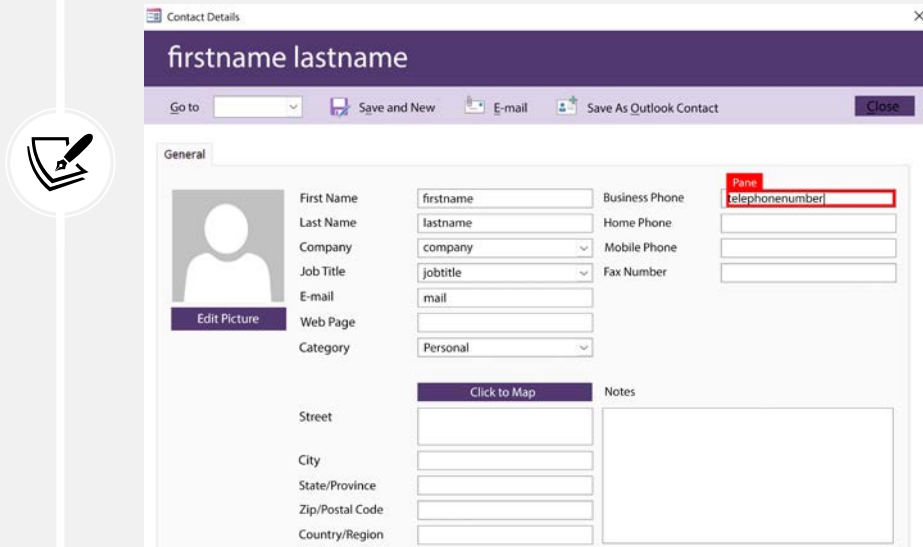


Figure 16.24: The Power Automate recorder

4. Click the **New Contact** button.
5. Add placeholder data to the **First Name**, **Last Name**, **Company**, **Job Title**, **E-mail**, and **Business Phone** number fields. Click to select each field:

Every keypress is recorded, so be judicious with your keystrokes. You only need basic placeholder information, since we're going to edit the steps and replace the text entered with variables. You only need to have enough text to help you remember which field is which. Go slow enough so that the recorder captures your clicks on the correct fields. Notice the red highlighting around fields as you hover and select them. This is critical data that the recorder is capturing.



The screenshot shows the 'Contact Details' form in Microsoft Access. The form has a purple header bar with the text 'firstname lastname'. Below the header is a toolbar with buttons for 'Go to', 'Save and New', 'E-mail', 'Save As Outlook Contact', and 'Close'. The main area is divided into two sections: 'General' and 'Notes'. The 'General' section contains fields for 'First Name', 'Last Name', 'Company', 'Job Title', 'E-mail', 'Web Page', 'Category', 'Business Phone', 'Home Phone', 'Mobile Phone', and 'Fax Number'. The 'Notes' section has a large text area. A red box highlights the 'Business Phone' field, which contains the text 'telephonenumber'.

Figure 16.25: Adding data to the contact entry

6. On the **Contact Details** page, click the **Close** button.
7. Click the **Close** button to exit Microsoft Access.

8. Click **Finish** on the Power Automate recorder:

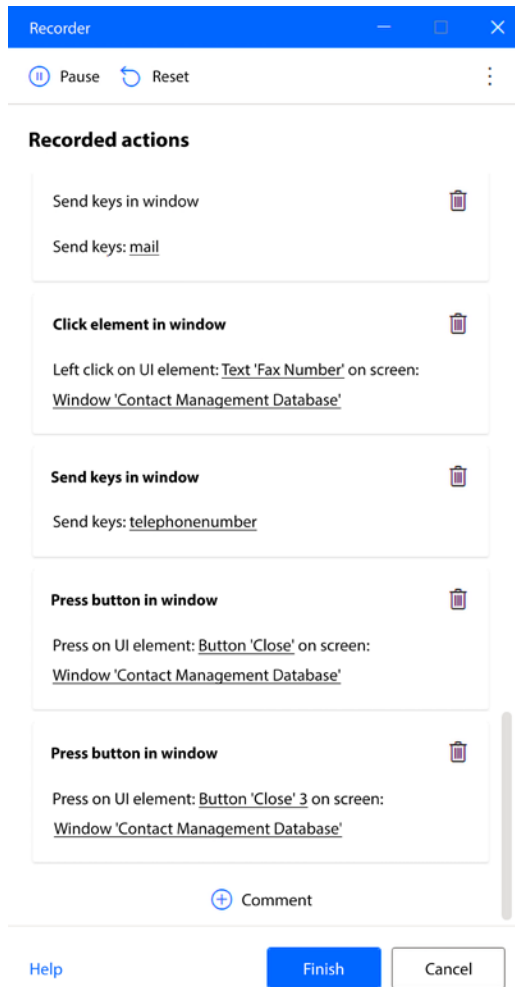


Figure 16.26: Finishing the step recording

9. You should be returned to the Power Automate Desktop designer interface. The main window displays all the actions that you took while editing the contact form. Carefully review the actions listed. If you see an action that is out of place or in any way would interfere with the flow running in an automated fashion (such as closing another application, moving a window, or clicking in an unrelated field), you can select the action and delete it by either pressing the Delete key on the keyboard or expanding the ellipsis and clicking **Delete**:

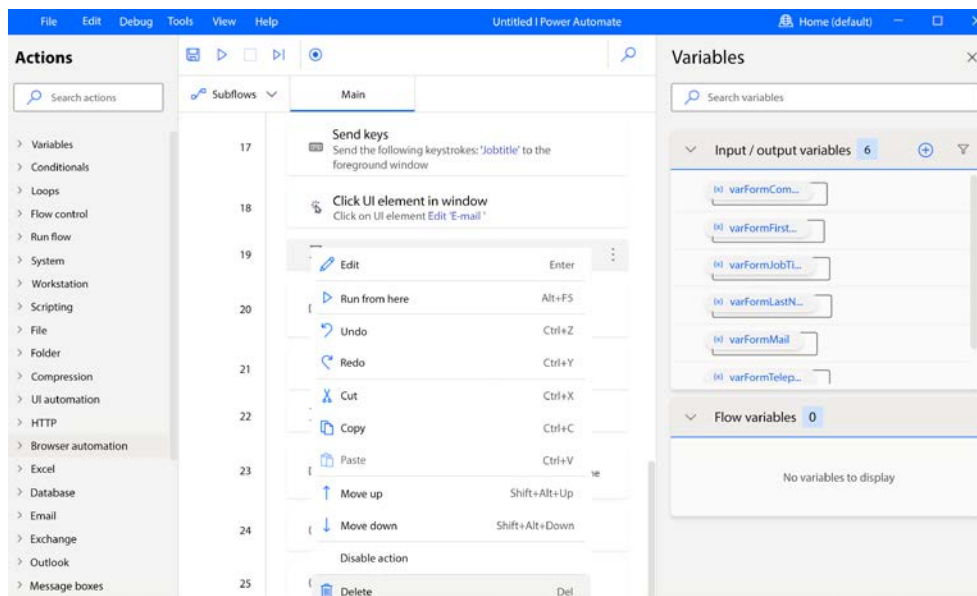


Figure 16.27: Deleting actions

10. After deleting any errant items, walk through the final set of steps to ensure they logically make sense and that all of the steps have been accounted for. If you accidentally delete or change the order of steps, you can press *Ctrl + z* to backstep through your actions. While you're reviewing the steps, take note of the step numbers with the **Send keys** actions, since those will contain your placeholder text.
11. Next, we'll replace the placeholder text you entered during the recording process with the correct variable values.

## Updating the variables

As you examined the steps, you should have seen the textboxes (such as **First Name**) that you selected and entered your placeholder text into. In this section, we're going to go through the process of replacing the placeholder text with the input variables you created during the *Configuring the flow framework* section:



- 1. Locate the first instance of a **Send keys** action, which should correspond to you entering the placeholder text value into the **First Name** field on the **Contact Details** form during the recording process:

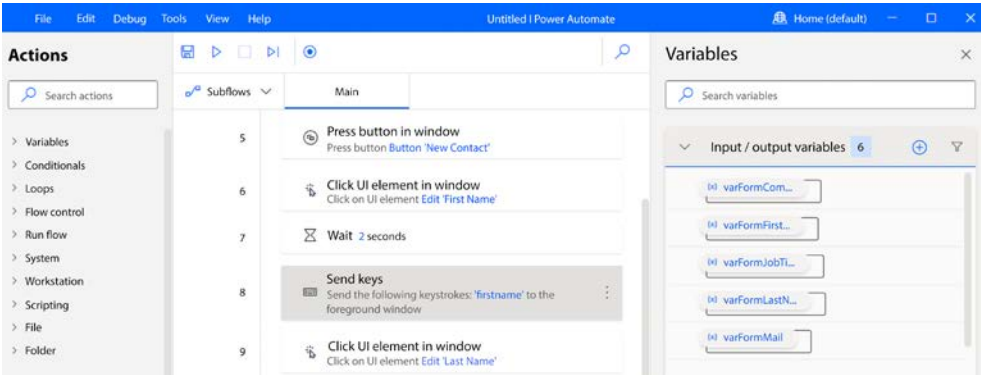


Figure 16.28: Locating the First Name field

- 2. Double-click the entry.
- 3. In the **Send keys** details window, select text placeholder text (in this example, the value `firstname`):

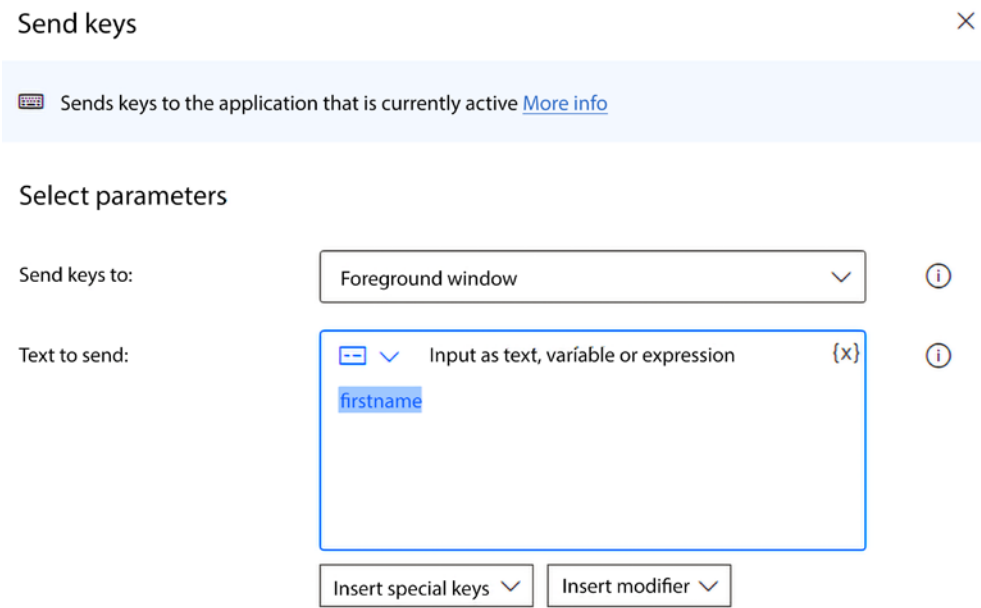


Figure 16.29: Editing the Send keys data

4. In the upper-right area of the **Text to send** textbox, choose the **Select variable** icon, represented by **{x}**:

Send keys ×

Sends keys to the application that is currently active [More info](#)

Select parameters

Send keys to: Foreground window ⓘ

Text to send:  ⌵ Input as text, variable or expression {x} ⓘ

Invalid value

Search variables

Delay between keystrokes: Name Type ⌵ ⓘ

Send Text as hardware keys: ⌵ Input /output variables 6 ⓘ

On error

>	varFormCompa...	Text value
>	varFormFirstNa...	Text value


Cancel

Figure 16.30: Choosing the variable selector

5. From the list of input variables displayed, select the variable representing the **First Name** value. In this case, the variable is named **varFormFirstName**. Click **Select** to confirm your choice.


6. Verify that the data in the **Text to send** box has been updated with %varFormFirstName%, as shown in *Figure 16.31*:

### Send keys ✕

 Sends keys to the application that is currently active [More info](#)

#### Select parameters

Send keys to: Foreground window ⌵ i

Text to send:  ⌵ Input as text, variable or expression {x} i

%varFormFirstName%

Insert special keys ⌵ Insert modifier ⌵

Delay between keystrokes: 10 ⬆ ⬇ ⬆ i

Send Text as hardware keys: ☐ i

---


 On error Save Cancel

Figure 16.31: Confirming the correct variable data

7. Click **Save**.
8. Repeat this series of steps for the remaining variables corresponding to last name, company, job title, email address, and telephone number.
9. Click **Save** in the Power Automate designer.

10. Exit the Power Automate designer.

Congratulations! You've recorded a flow. Now, we'll go back to the Power Automate web portal and finish up the flow.

## Finishing the flow

Now that you've got all of the input variables created in the desktop flow, it's time to update the cloud flow to send the Microsoft Forms data dynamic content tokens to their corresponding desktop flow input variables.

Let's go!

1. With the **Add Contact to Access Database** flow open in the browser, click **Save**.
2. Click **Refresh** on the browser toolbar. This should reload the variables from the desktop flow, as shown in *Figure 16.32*:

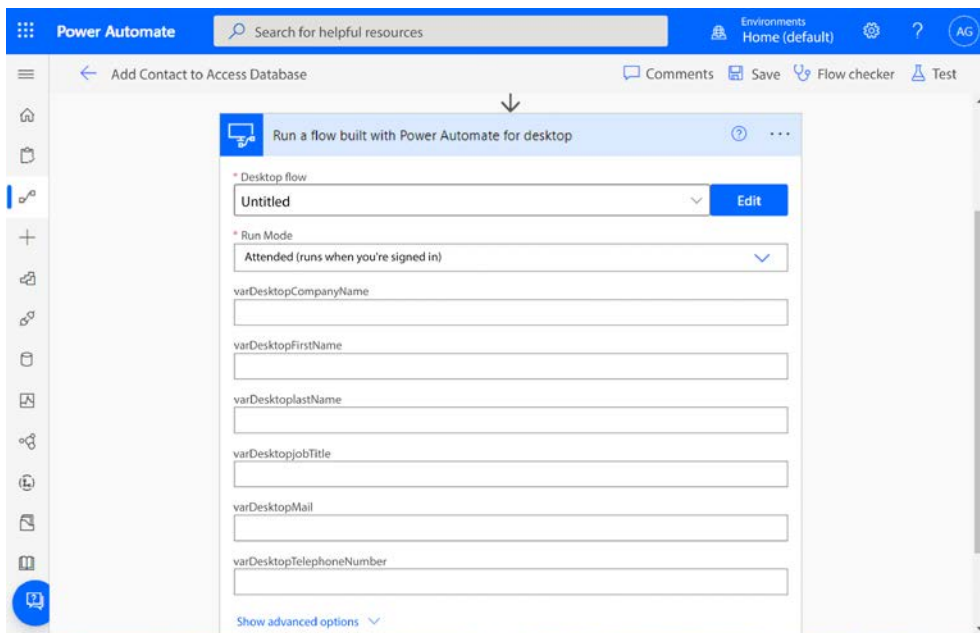


Figure 16.32: Editing the flow to update variables

3. Starting with the first input variable listed, click in the textbox and select the dynamic content token from the **Get response details** action that corresponds to the variable:

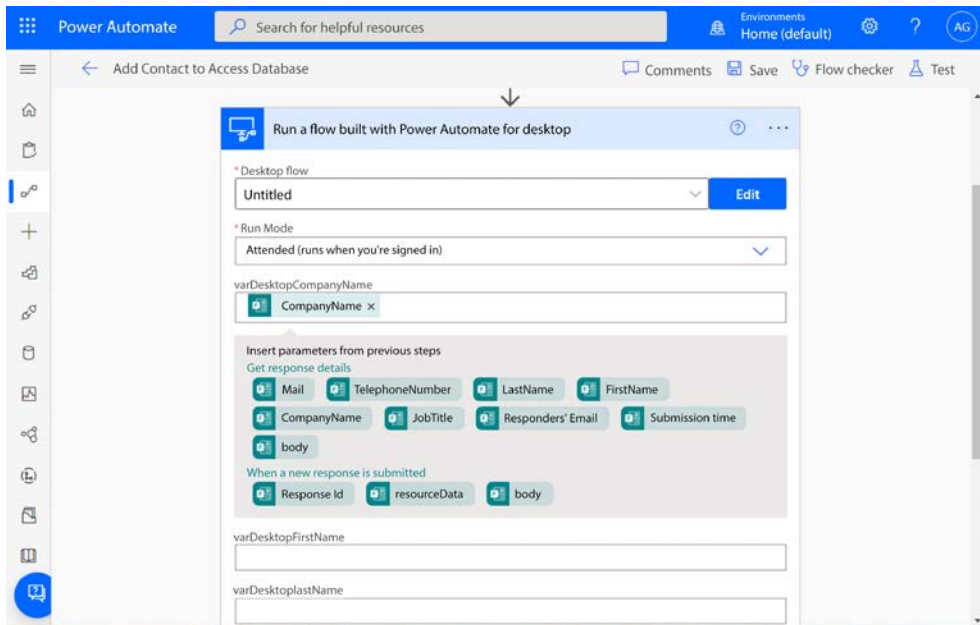


Figure 16.33: Linking dynamic content to corresponding desktop input variables

4. Repeat this process for each variable.
5. Click **Save** when finished.
6. Close the web browser window and ensure no other applications are running on the machine that will be executing the desktop flow.

You've finished configuring a desktop flow. Now, on to testing it.

## Testing the flow

As mentioned in the *Configuring prerequisites* section, it is recommended to initiate this flow on a device separate from the computer that will be executing the desktop flow. If possible, you'll want to have access to both devices at the same time, so you can start the flow on one device and then watch the desktop flow execute on the second device. This will help you gather any data necessary for troubleshooting issues.

To test the flow, follow these steps:

1. On a secondary computer, open a web browser.

2. Navigate to the Power Automate web portal (<https://flow.microsoft.com>).
3. Click **My flows**, select the **Add Contact to Access Database** flow, and click **Edit**.
4. Click **Test** to open the **Test Flow** panel.
5. Select the **Manually** radio button, and then click **Save & Test**.
6. Open another browser tab and navigate to the Microsoft Forms portal (<https://forms.microsoft.com>). Select the **Customer Survey** form:

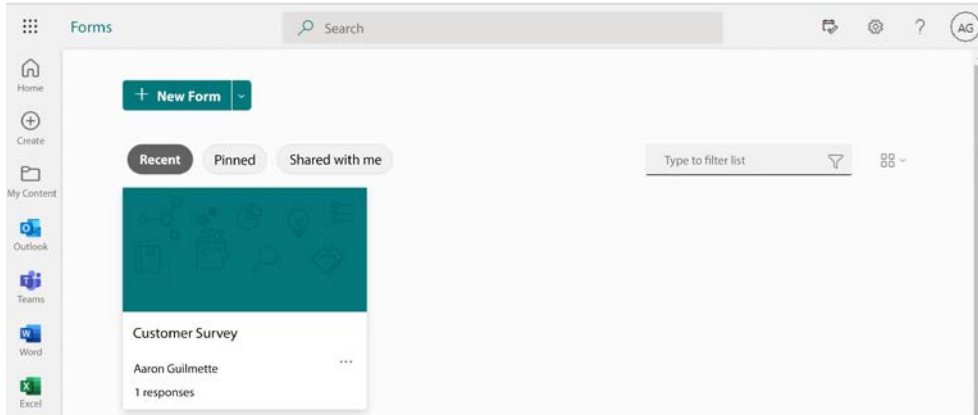


Figure 16.34: Choosing the Customer Survey form

7. On the menu bar, select **Preview**:

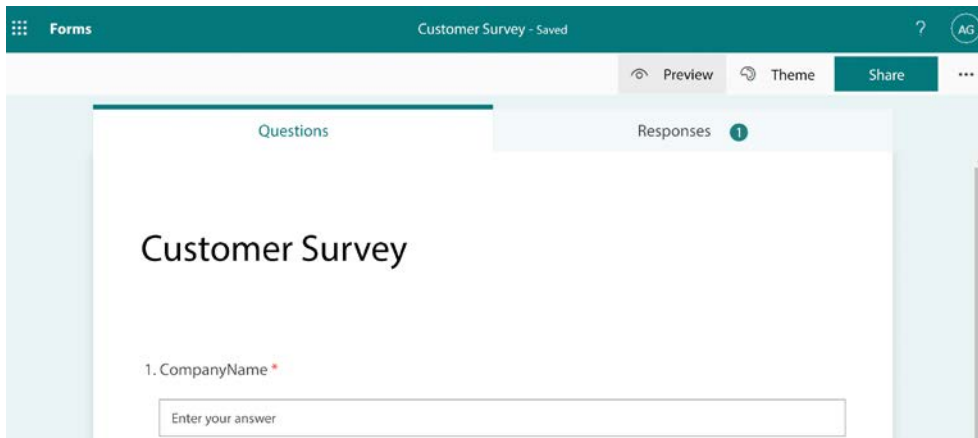


Figure 16.35: Previewing the form

8. Fill out the form with unique and easily identifiable test data and then click submit.

At this point, you can switch back to the Power Automate browser window and review the output of the steps.

## Verifying the flow

The easiest way to verify a successful desktop flow completion is to watch the desktop flow execute in real time. During the execution process, you should have been able to watch the following actions take place:

- The Microsoft Access application launch and open the Contacts database
- The **New contact** button being pushed
- Text being entered in each of the fields
- The **close** button being pushed
- The Microsoft Access application close

You can also verify the flow by opening the Contact database yourself and confirming that the data matches what you entered in the Customer Survey form:

The screenshot shows the 'Contact Details' form in Microsoft Access. The title bar reads 'Contact Details' with a close button. The form has a purple header with the name 'John Doe'. Below the header is a toolbar with buttons: 'Go to' (with a dropdown), 'Save and New', 'E-mail', 'Save As Outlook Contact', and 'Close'. The 'General' tab is selected. On the left is a placeholder for a profile picture with an 'Edit Picture' button. To the right of the picture are fields for: First Name (John), Last Name (Doe), Company (Contoso), Job Title (Manager), E-mail (johnd@contoso.com), Web Page, and Category (Personal). Further right are fields for Business Phone (4255551212), Home Phone, Mobile Phone, and Fax Number. At the bottom left are fields for Street, City, State/Province, Zip/Postal Code, and Country/Region. A 'Click to Map' button is positioned above the Street field. On the bottom right is a large 'Notes' text area.

Figure 16.36: Viewing a contact that was entered through the Power Automate Desktop flow

Finally, you can review the Power Automate run history for the flow:

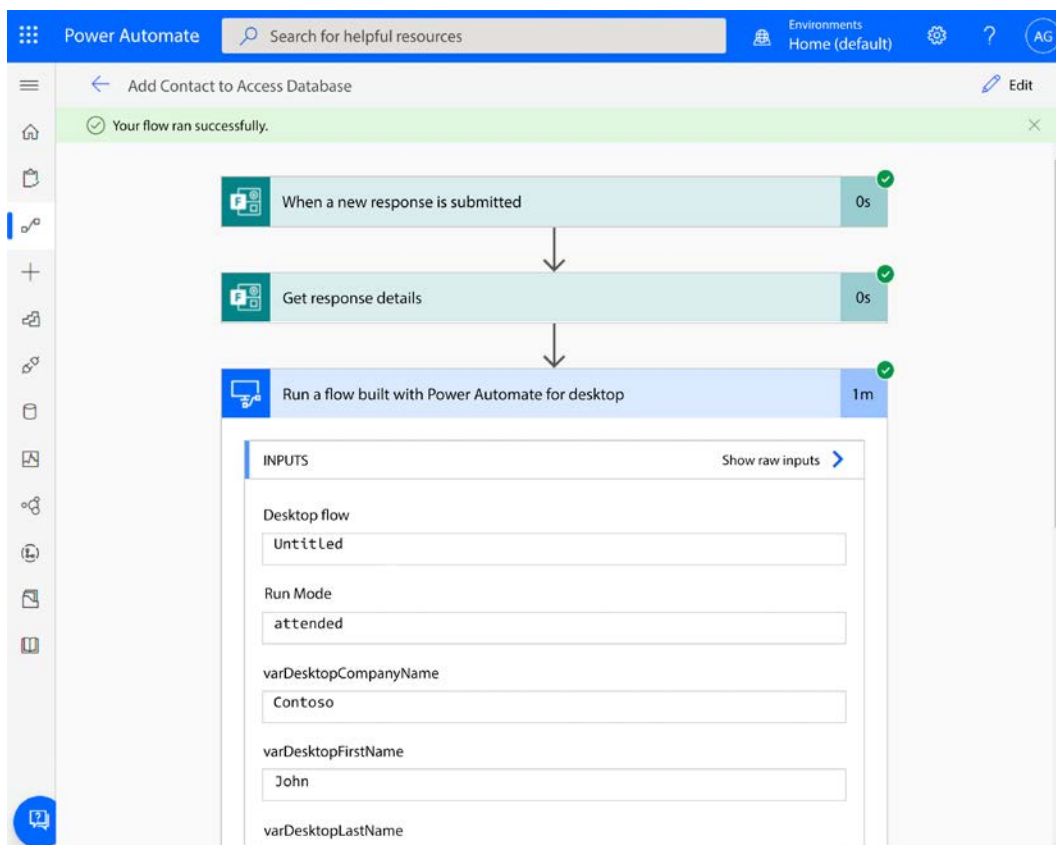


Figure 16.37: Verifying data in the flow's run history

Any of these methods confirm that your flow was created and executed. If you encounter errors or if data appears in the wrong locations you can examine the steps for output information, including networking or credential errors, or the wrong textboxes chosen for inserting variable data.

You've successfully automated an application using RPA!

## Summary

RPA has the potential to help streamline operations for legacy applications that weren't previously candidates for automation. This is incredibly beneficial for organizations that rely on older applications that are not network-aware or use older communications protocols that can't easily be routed over the internet.



By following the examples in this chapter, you learned how to integrate a cloud flow with Power Automate Desktop, record steps in an application, and pass variables between environments – all of which are useful in crafting desktop flows specific to your needs. By being able to pass variables into the Power Automate Desktop environment, you can simulate the process of a human interacting with an application, adding value in ways not possible before.

In the next chapter, we're going to learn about using AI models with Power Automate.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 17

## Introducing AI Models

Throughout the examples in this book, you've used conditions to determine which branch of a flow should execute. You've used conditions like:

- Is the purchase request value more or less than \$1,000?
- Does the user have more or less than 100 Twitter followers?
- Is the approval response equal to *Approved*?

Those types of evaluations are useful when you're working with hard data – numbers, Boolean values, or exact string matches. But what happens when you're trying to make decisions with less straightforward data?

That's where **Artificial Intelligence (AI) models** can become useful. AI models use AI to make determinations about a variety of different content and data types. For example, it may be very easy for a human to read a business card and determine what value is the individual's name and what value is the individual's job title, regardless of where they are placed on the card. This task, however, is much more difficult for a computer to accomplish. As business card designs change, data fields move and become more difficult for a traditional automation engine to recognize and identify. AI models help bring contextual awareness to data, making it easier for automation to perform like a person.

Microsoft Power Automate includes several prebuilt AI models, as well as the ability to train and customize several additional model types. Prebuilt models focus on AI that's pre-trained with a particular data set to do a particular type of task, while custom models require a lot more configuration and training with your own data sets.

In this chapter, we’re going to focus on using the prebuilt AI sentiment analysis model to evaluate whether feedback received via a Microsoft Forms survey is positive or negative. The topics we’ll cover include:

- Learning about AI models
- Technical requirements
- Configuring prerequisites
- Creating a sentiment analysis flow

Let’s get started!

## Learning about AI models

There are many types of models available in the Power Platform, covering a range of data types and scenarios. You can use models to analyze text, scanned documents, images, email, or nearly any other type of input you can present to the service.

The data in *Table 17.1* describes the type of content that a model is designed to work with, the name of the model, the build type of the model (whether it’s a prebuilt or custom model) and a sample business use case.

Model name	Data type	Build type	Business case
Business card reader	Documents	Prebuilt	Business card scanning
Form processing	Documents	Custom	Job application processing
Text recognition	Documents	Prebuilt	Identify text in a set of crime scene photos
Receipt processing	Documents	Prebuilt	Automating expense reports
Category classification	Text	Prebuilt or custom	Categorize focus group feedback
Entity extraction	Text	Prebuilt or custom	Extract insights from product or service reviews
Key phrase extraction	Text	Prebuilt	Get alerts on news items or social media posts referencing a brand
Language detection	Text	Prebuilt	Identify the language of submitted text

Model name	Data type	Build type	Business case
Sentiment analysis	Text	Prebuilt	Classify positive and negative customer feedback
Text translation	Text	Prebuilt	Translate email support requests into the support engineer's language
Prediction	Structured data	Custom	Alert on fraudulent transactions
Image classification	Images	Custom	Identify inappropriate content in customer -uploaded images
Object detection	Images	Custom	Quickly identify the type of machine on a shop floor by taking a picture of it

Table 17.1: AI Model types

Each model has its own set of strengths. Depending on the business goals, you may even need to use more than one AI model to complete processing from end to end.

In this chapter, we’re going to focus on using the prebuilt sentiment analysis model. Sentiment analysis reviews the text content and returns one of four ratings: positive, negative, neutral, or mixed. Each sentence is analyzed individually, and the model returns a sentiment value for each sentence as well as the content as a whole.

In addition to assessing the sentiment of the content (both the individual and total), sentiment analysis also provides a confidence score on a scale of 0 to 1. Values closer to 1 indicate increasing confidence in the accuracy of the analysis. There are a number of factors that can influence confidence, such as improper or incorrect word usage.

For the purposes of this example, we’re going to focus on the overall sentiment and confidence. Next, we’ll focus on the technical requirements for using AI model-based solutions.

## Technical requirements

AI models (managed through AI Builder in the Power Platform) require access to the Microsoft Dataverse. If you’ve completed examples earlier in this book that required access to Dataverse, then you should have already met the requirements. If you haven’t created a Dataverse database yet, you can use the following process:

1. Navigate to the Power Automate web portal (<https://flow.microsoft.com>).
2. Expand **AI Builder**, and then select **Explore**:

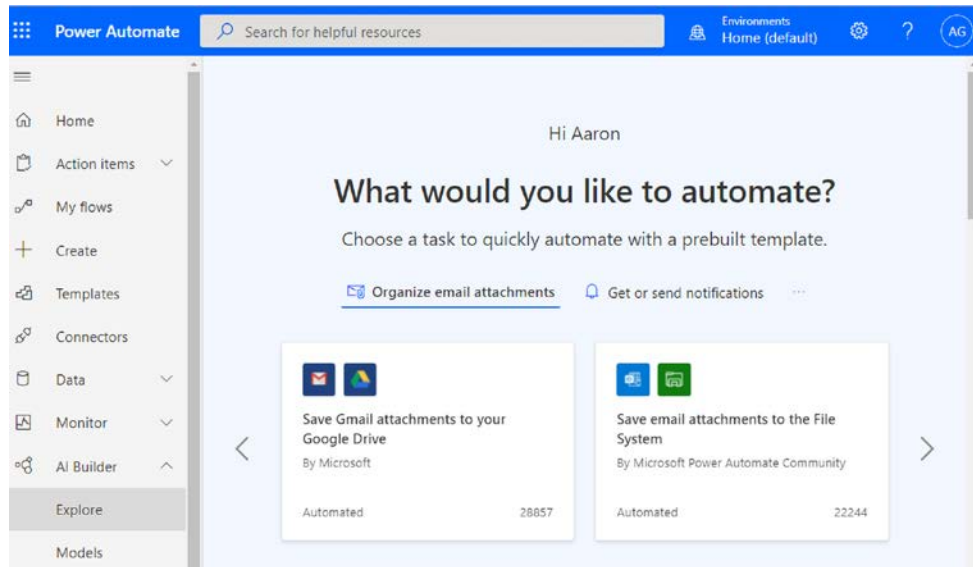


Figure 17.1: Launching AI Builder

3. If prompted, select **Create a database**.
4. Select a **Currency** and a **Language**, and then select **Create my database**:

**New database** ✕

Choose the currency and language your data should use. ⓘ

**Currency** ⓘ

USD ▾

**Language** ⓘ

English ▾

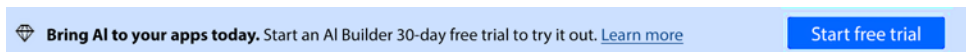
☒ Include sample apps and data

ⓘ By choosing Create my database, you agree Microsoft can use table and column names that you create (but not content in the database tables) to help improve our common data model. These names may be stored in our diagnostic systems and copied across regions. [Learn more](#)

Cancel Create my database

Figure 17.2: Creating the Dataverse database

5. Use your browser's refresh button to update the page.
6. At the top of the screen, click **Start free trial**:



## Add Intelligence to your business

Automate processes, find insights, and make applications more productive. Choose from prebuilt and customizable AI models, or leverage the models your company has already built. [Learn more](#)

Figure 17.3: Enabling a trial for AI Builder

Once the trial has successfully started, you can start working with AI models.

In addition to Dataverse capability, you also need AI Builder capacity. AI Builder capacity is licensed separately, though there is also some capacity included with *Power Apps per app* plans, *Power Apps per user* plans, and *Power Automate per user with Attended RPA* plans. If you have already subscribed to a trial license such as the *Power Automate per user with Attended RPA* plan, you should have enough capacity to begin learning about AI models.

For more information on licensing and capacity, see <https://docs.microsoft.com/en-us/ai-builder/administer-licensing>.

## Configuring prerequisites

As mentioned at the beginning of the chapter, the goal of this flow is to analyze customer feedback received through a form. In order to do this, we’re going to configure two things:

- A Microsoft Forms survey to collect data such as a customer’s contact information, a product model, a product serial number, purchase date, and a text area where they can write their product feedback
- An Excel spreadsheet in a SharePoint document library to store the form response data and sentiment analysis for later review

If you don’t want to create new SharePoint sites or forms, you can use existing forms that you’ve previously created but will have to adapt the flow accordingly.

## Creating a Forms survey

The Forms survey will be used to capture the feedback. In this example, we’re going to capture a lot of data to emulate how something like this would look if an organization were to deploy it.

Here are the important data points to capture:

Field name or question	Data type	Comments or restrictions
First Name	Text	
Last Name	Text	
Address 1	Text	
Address 2	Text	
City	Text	

Field name or question	Data type	Comments or restrictions
State	Text	You can also experiment with a choice/drop-down selector to force data quality
ZIP	Text	Restrictions: Number
Phone	Text	Restrictions: Number
Email	Text	
Contact Preference	Choice	Email, Phone, Text
Product Model	Choice	Widget 2000, Widget 2001, Widget 2002
Product Serial Number	Text	
Purchase Date	Date	
Feedback Value	Text	Enable the Long answer option

Table 17.2: Forms data fields and types

If your locality has different formats for address or phone values, you may wish to change your input restriction data.

To begin creating the form, follow these steps:

1. Navigate to Microsoft Forms (<https://forms.microsoft.com>) and select **New Form**.
2. Click the text **Untitled form** and replace it with a meaningful name, such as Customer Feedback Survey.
3. Click **Add new** to begin adding question fields, using the data in *Table 17.2* as a guide.
4. Click **Share** and select **Anyone can respond**:

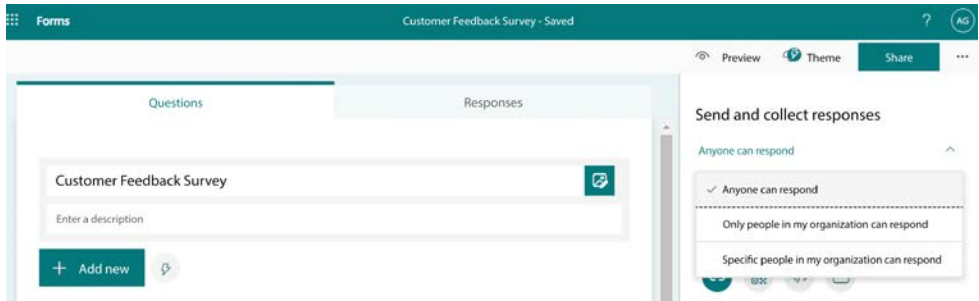


Figure 17.4: Enabling external access to form



5. Click **Copy** to copy the URL value and save it to a temporary location, such as Notepad:

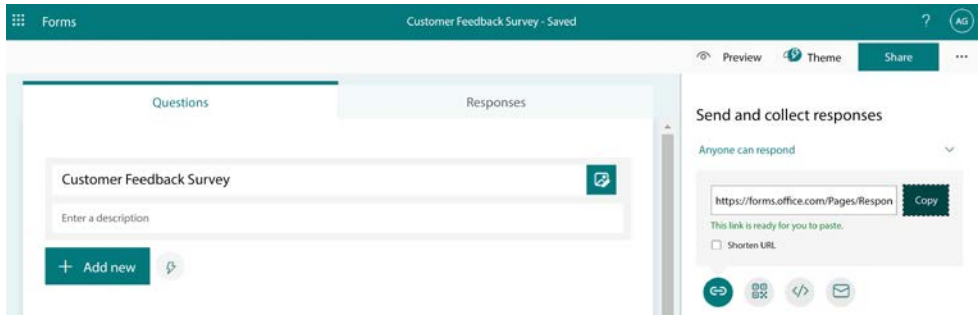


Figure 17.5: Saving the URL

Preview the form to ensure the fields are laid out correctly and that the data validation parameters you configured (if any) are working as anticipated.

Next, we’ll move on to creating an Excel workbook to store the results.

## Creating an Excel workbook

This Excel workbook will be used to store the submitted form’s results, as well as other data generated during the flow (such as the date and the sentiment analysis value). This workbook will also feature a table, which will allow you to send specific data fields from the form response into specific columns in the table.

To create the workbook, follow these steps:

1. In a new or existing SharePoint site, navigate to the default document library.
2. Click **New** and select **Excel workbook**:

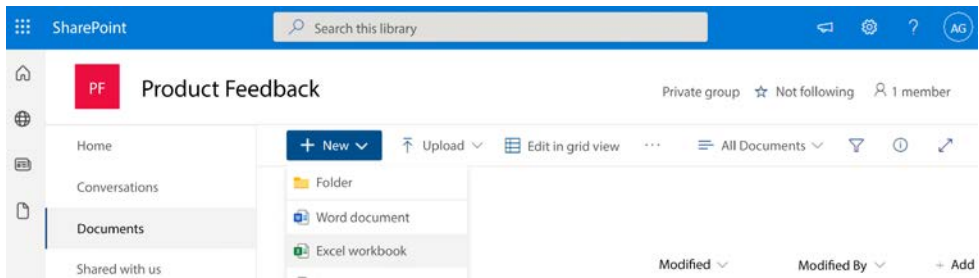


Figure 17.6: Creating a new Excel workbook

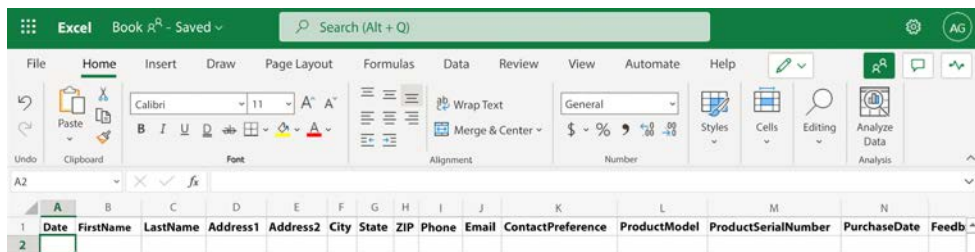
3. Populate the columns with the following values:

- Date
- FirstName
- LastName
- Address1
- Address2
- City
- State
- ZIP
- Phone
- Email
- ContactPreference
- ProductModel
- ProductSerialNumber
- PurchaseDate
- FeedbackValue
- SentimentAnalysisResult
- SentimentConfidenceLevel



*Ensure that none of the column names have spaces, as it may cause issues with the connectors or actions.*

4. Review the workbook, as shown in *Figure 17.7*:



*Figure 17.7: Configuring the Excel workbook*

5. Select all rows and columns.
6. Click **Insert** on the Excel ribbon, and then select **Table**:

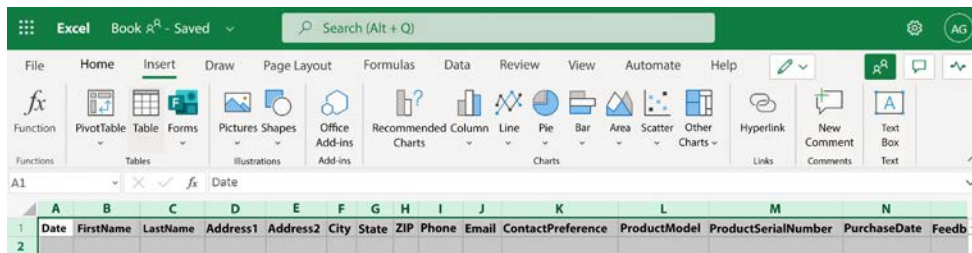


Figure 17.8: Selecting the rows and columns

7. Select the **My table has headers** checkbox and then click **OK**:

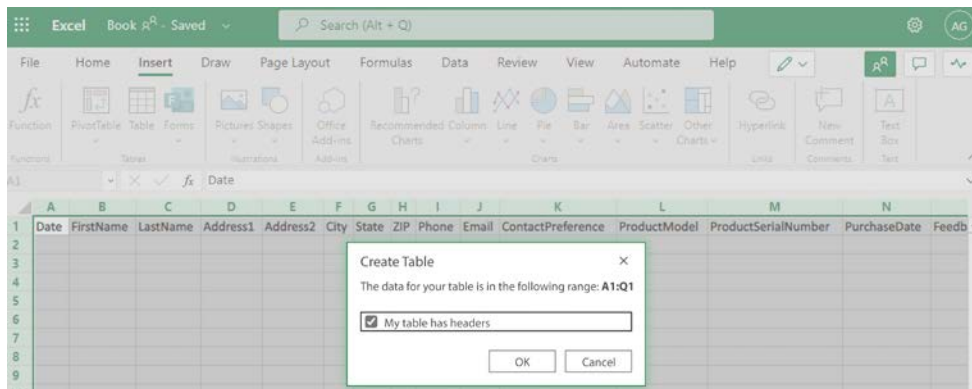


Figure 17.9: Creating a table

Once you've configured the workbook with a table, you're ready to begin creating the flow.

## Creating a sentiment analysis flow

By this point, we've configured mechanisms for data capture and storage. Now, we'll turn to analyzing the data. To start building the flow, follow these steps:

1. Navigate to the Microsoft Power Automate web portal (<https://flow.microsoft.com>) and click **Create**.
2. Select **Automated cloud flow**.

3. Enter a name and choose the **When a new response is submitted** Microsoft Forms trigger, as shown in *Figure 17.10*:

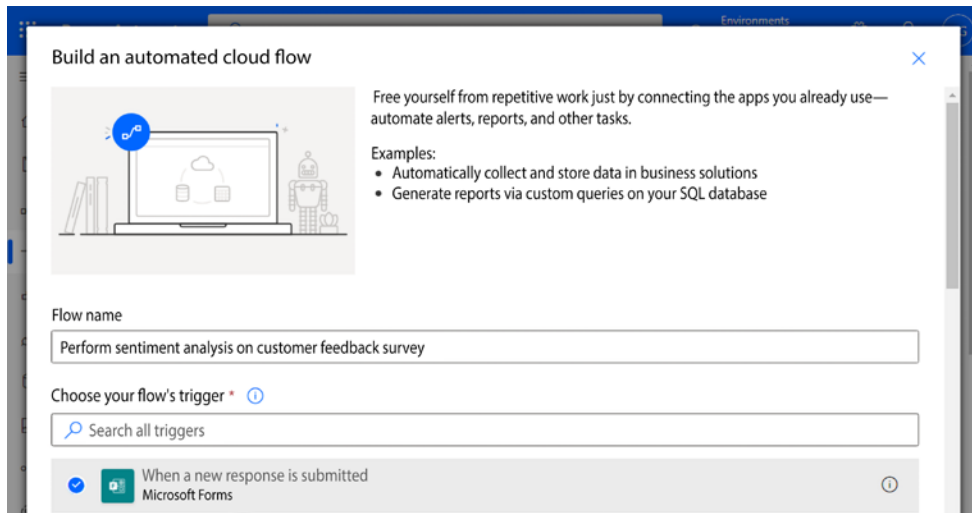


Figure 17.10: Creating a new automated cloud flow

4. Click **Create**.
5. In the **When a new response is submitted** trigger, select the **Customer Feedback Survey** form that you created during the prerequisites. If you don't see your form listed, you can select **Enter custom value** and use the ID value (the highlighted portion after *id=*) that you captured at the end of the *Creating a Forms survey* section:



Figure 17.11: Extracting the form ID value

6. Click **New step**.
7. Select the **Get response details** Microsoft Forms action.
8. In the **Get response details** action, under **Form Id**, select the **Customer Feedback Survey** form. If the form is not displayed, select **Enter custom value** and manually enter the saved **Form Id** value.

9. Under **Response Id**, select the **Response Id** dynamic content token:

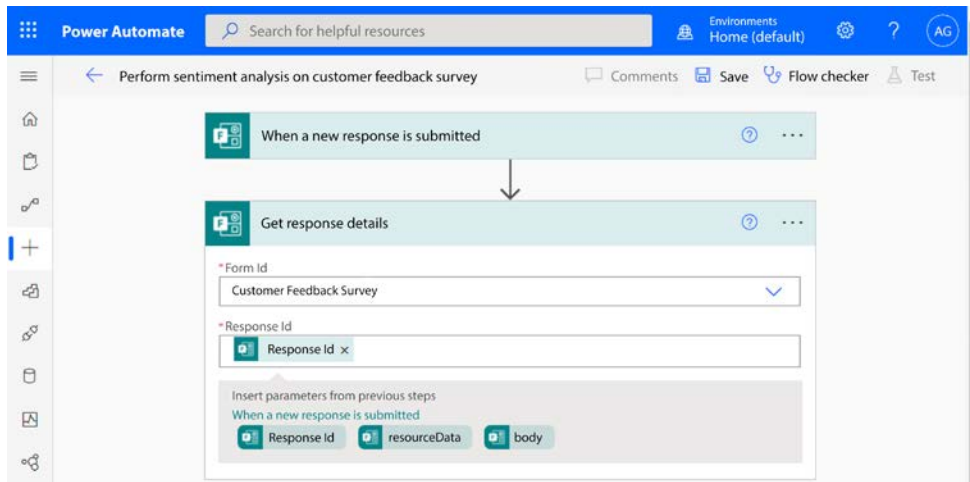


Figure 17.12: Configuring the Response Id

10. Click **New step**.
11. Select the **Detect the language being used in text** AI Builder action, as shown in Figure 17.13:

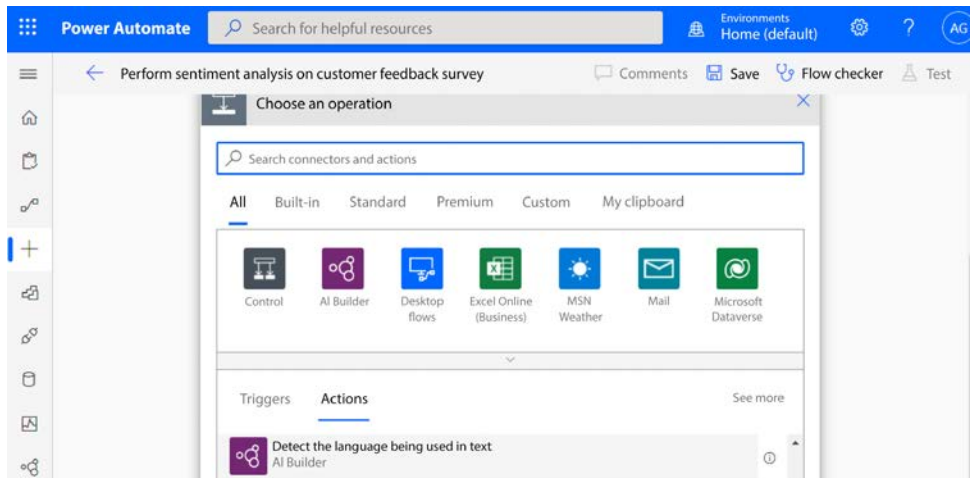


Figure 17.13: Adding a language detection action

You may receive a pop-up question at this point to start the free trial of AI Builder. You can find information about the licensing available in the *Technical requirements* section of this chapter.

12. Select the **Text** field in the **Detect the language being used in text** action, and then select the **Feedback Value** dynamic content value in the **Get response details** section, as shown in *Figure 17.14*:

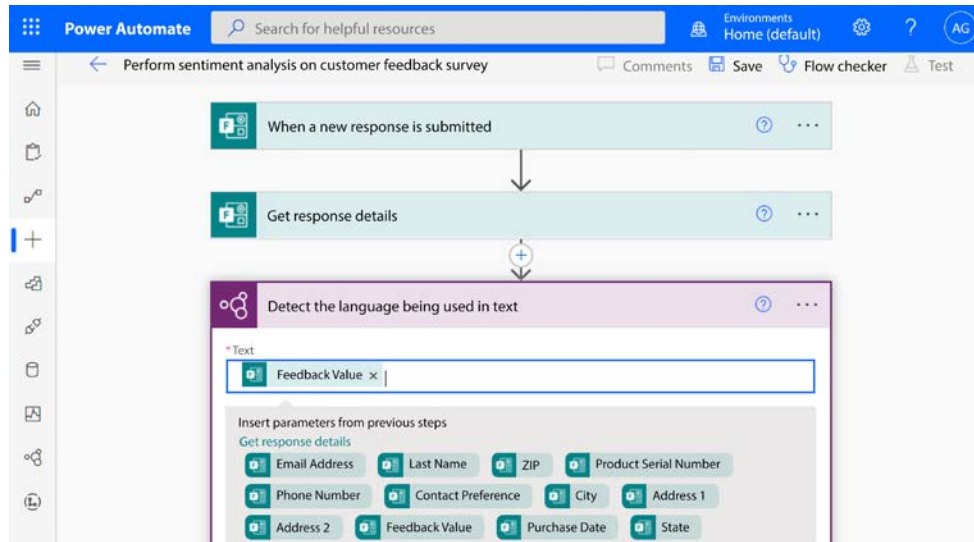


Figure 17.14: Configuring the language detection action

13. Click **New step**.
14. Select the **Analyze positive or negative sentiment in text** AI Builder action:

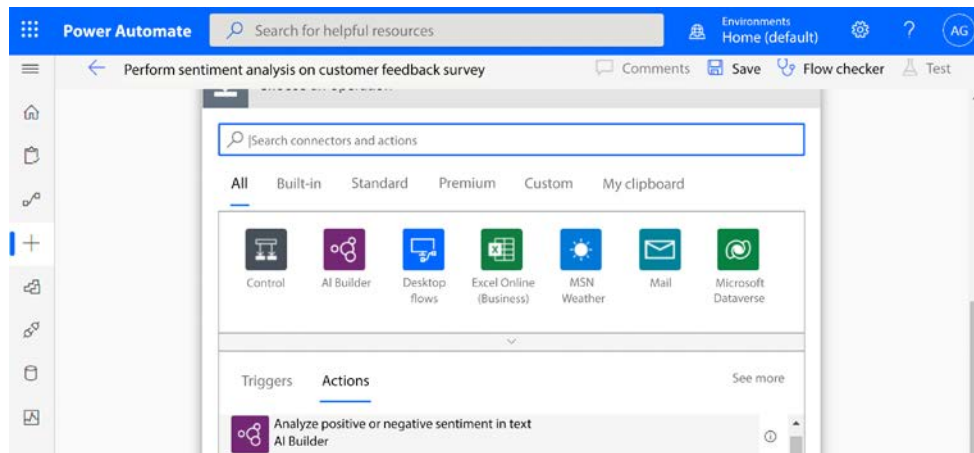


Figure 17.15: Adding the sentiment analysis action

15. In the **Language** dropdown of the **Analyze positive or negative sentiment in text** action, select **Enter custom value**:

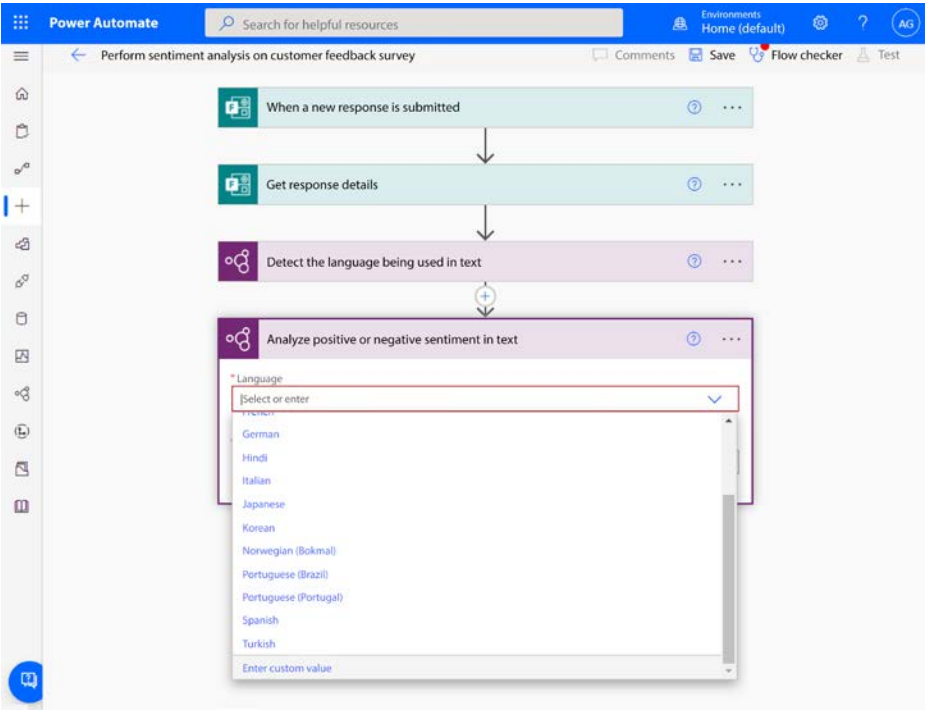


Figure 17.16: Configuring the language setting

16. Select the **Language** dynamic content token in the **Detect the language being used in text** category. The action will get wrapped in an **Apply to each** function:

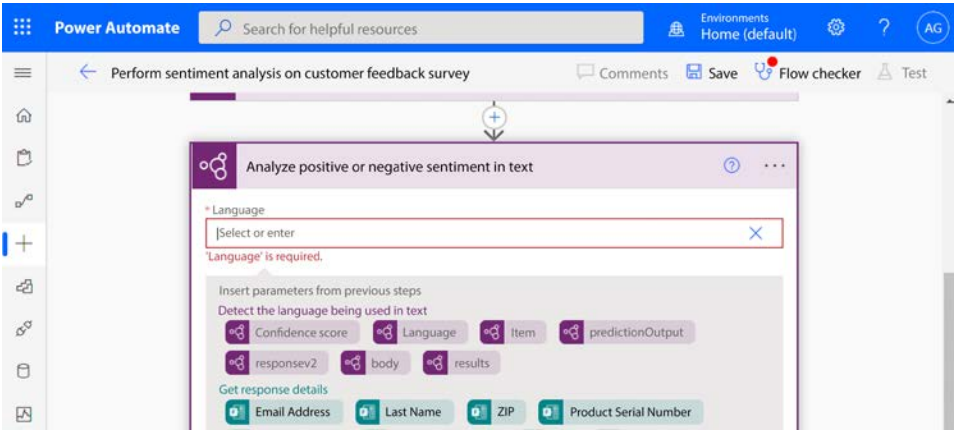


Figure 17.17: Selecting the language

17. Click the title bar for the **Analyze positive or negative sentiment in text** action inside the **Apply to each** function, and then select the **Text** field. Add the **Feedback Value** dynamic content token from the **Get response details** category, as shown in *Figure 17.18*:

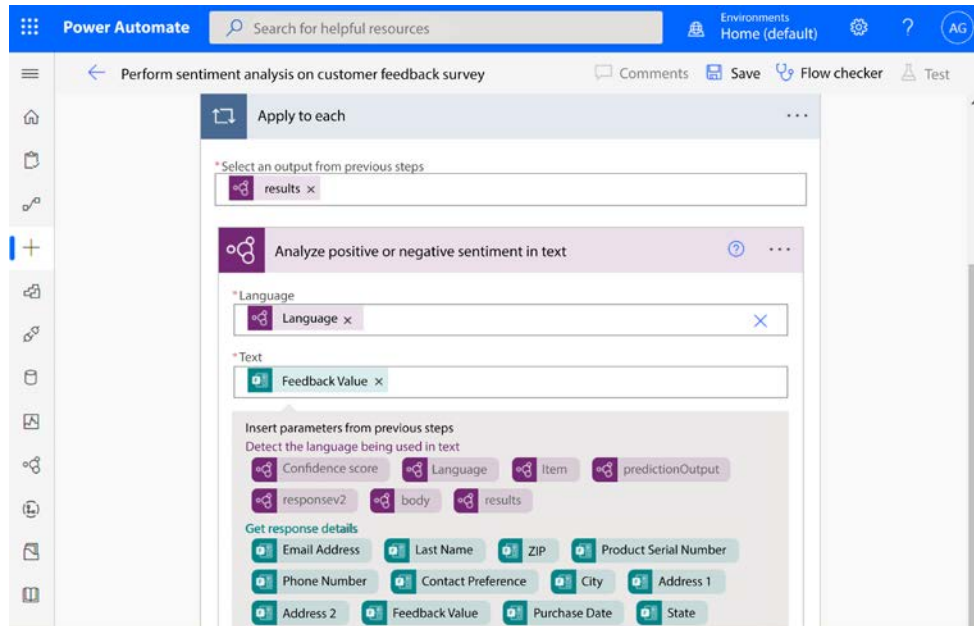


Figure 17.18: Configuring the sentiment analysis action

18. Inside the **Apply to each** wrapper, click **Add an action**, and then select the **Add a row into a table Excel Online (Business)** action:

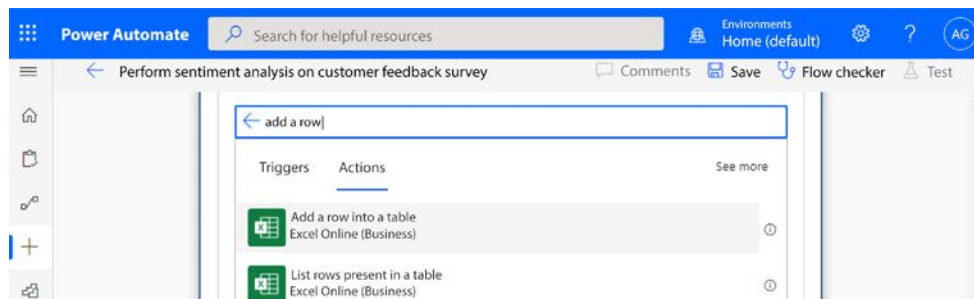


Figure 17.19: Selecting the Add a row action

19. Under **Location**, select the SharePoint site that contains the Excel workbook you created in the *Creating an Excel workbook* section.



20. Under **Document Library**, select the document library that contains the Excel workbook you created in the *Creating an Excel workbook* section.
21. Under **File**, use the file browser to select the Excel workbook you created in the *Creating an Excel workbook* section.
22. Under **Table**, select the table you created in the *Creating an Excel workbook* section:

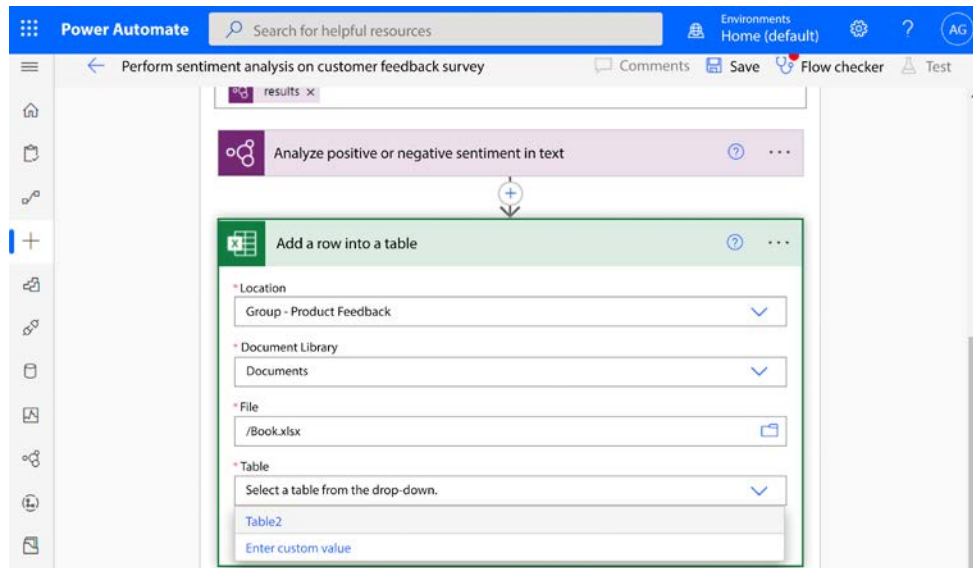


Figure 17.20: Selecting a table

23. Once the table has been selected, the **Add a row into a table** action should update to show a field for each column name configured in the table, as shown in *Figure 17.21*:

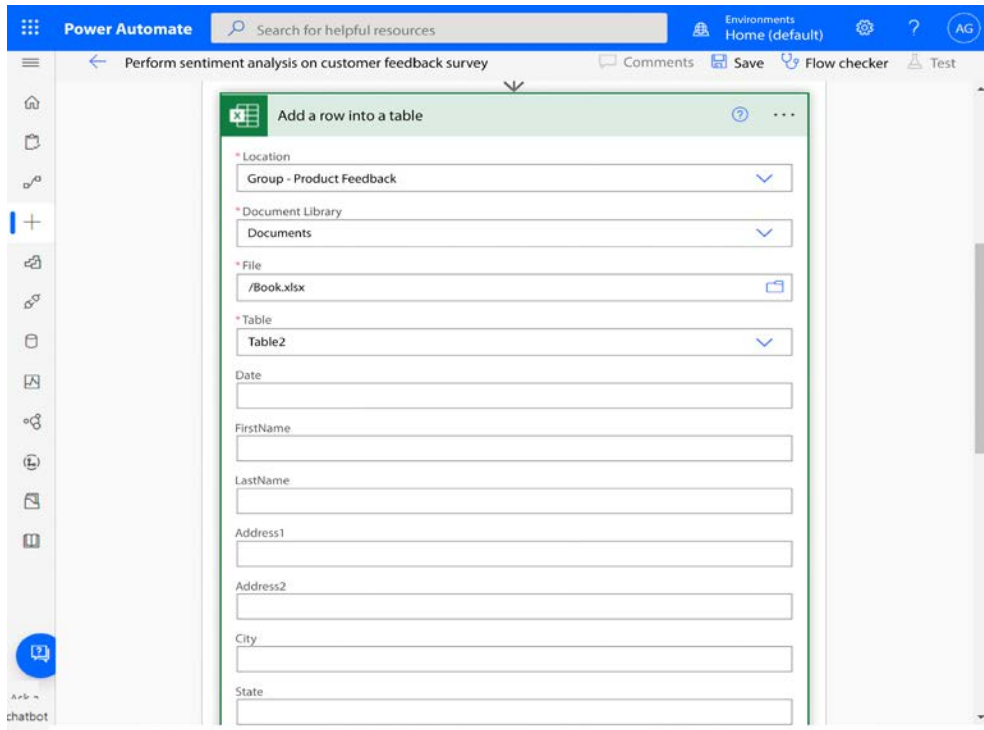


Figure 17.21: Fields added match the column names from the preconfigured table

24. In the **Date** field, select the **Submission time** dynamic content value in the **Get response details** category:

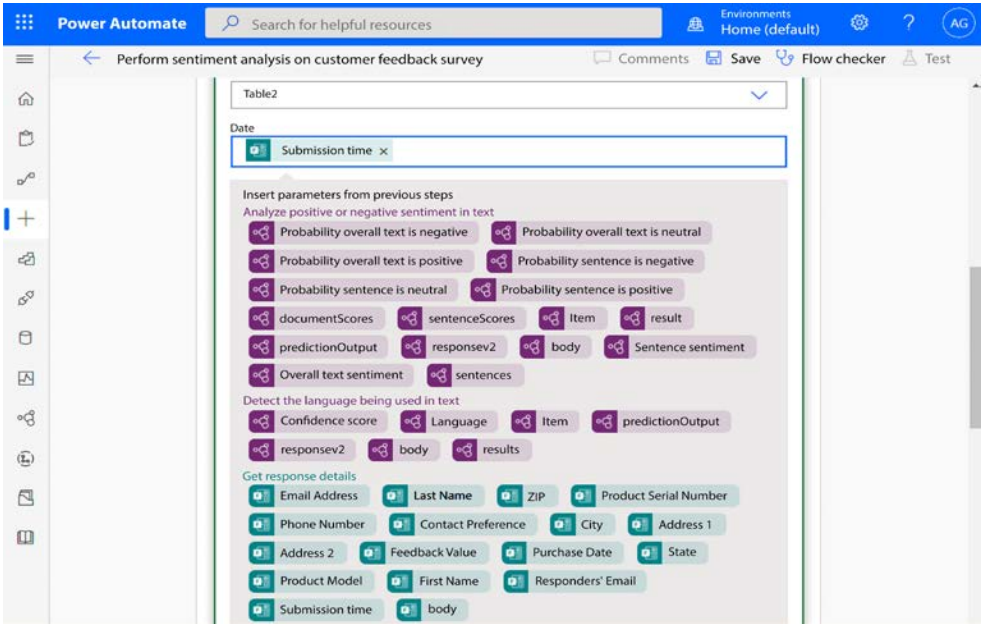


Figure 17.22: Adding Submission time to the Date value

25. Configure the action by matching the data fields you captured in the survey form (*FirstName*, *LastName*, *Address1*, *Address2*, *City*, *State*, *ZIP*, *Phone*, *Email*, *ContactPreference*, *ProductModel*, *ProductSerialNumber*, *PurchaseDate*, and *FeedbackValue*) with the corresponding **Get response details** dynamic content token. See Figure 17.23 for a completed example:

Be sure to use the dynamic content token value for **Email Address**, as opposed to the **Responder's Email** automatically generated value. The **Email Address** value that you captured during the form submission will contain the value that the respondent manually entered, while the **Responder's Email** value will contain the signed-in user's email address. If the responder isn't signed in, the email value may be blank.

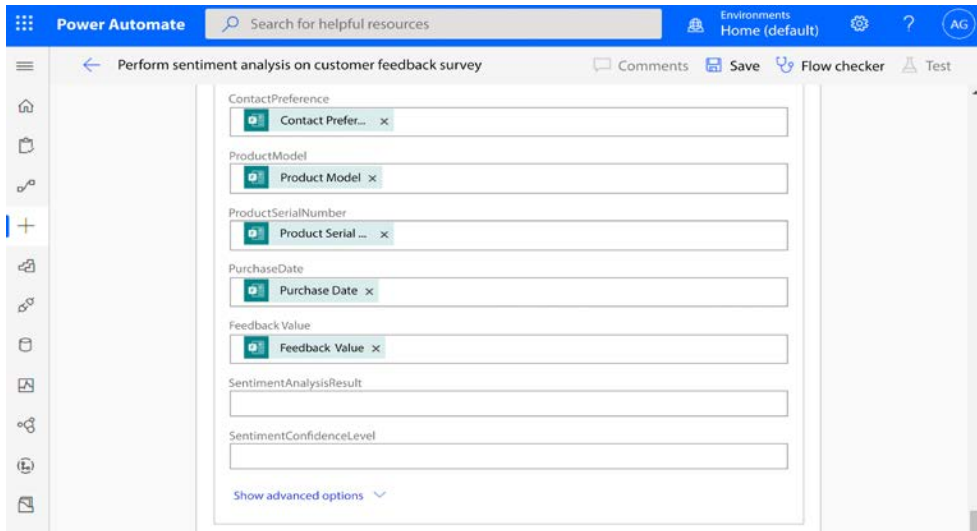


Figure 17.23: Validating the dynamic content token mapping

26. Select the **SentimentAnalysisResult** field. When the dynamic content values appear, select **Overall text sentiment** from the **Analyze positive or negative sentiment in text** category:

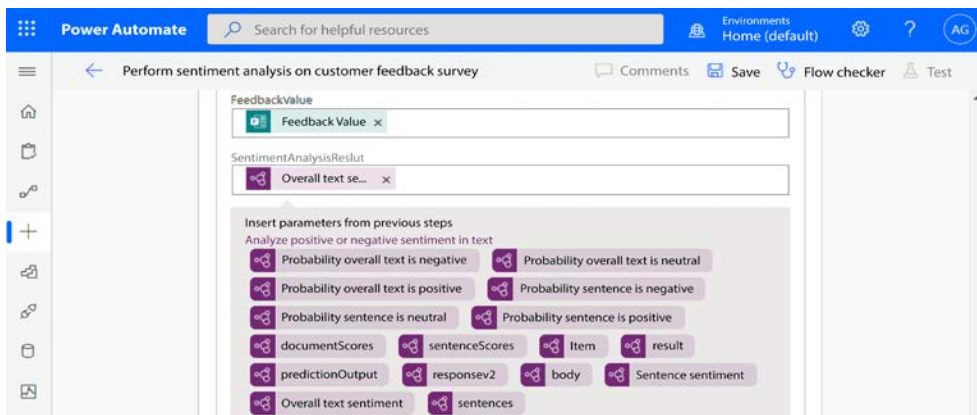


Figure 17.24: Adding the sentiment analysis result

27. Select the **SentimentConfidenceLevel** field and add the **documentScores** dynamic content value.
28. Click **Save**.

The sentiment analysis flow has been created. Now you're ready to begin analyzing surveys!

## Testing the flow

To test this flow, you'll need to submit several surveys with varying descriptions in the **Feedback Value** text area field. With each separate review, you'll want to use different phrasing to express what you think are negative or positive impressions of a product or experience.

Let's try a few:

1. From the flow designer page, click **Test**.
2. Select the **Manually** radio button and then click **Test**.
3. Open a new browser tab and navigate to the Microsoft Forms landing page (<https://forms.microsoft.com>). Select **Customer Feedback Survey** and then click **Preview**.
4. Fill out the survey. When you get to the **Feedback Value** text area, enter a value that you think indicates satisfaction or a positive experience:

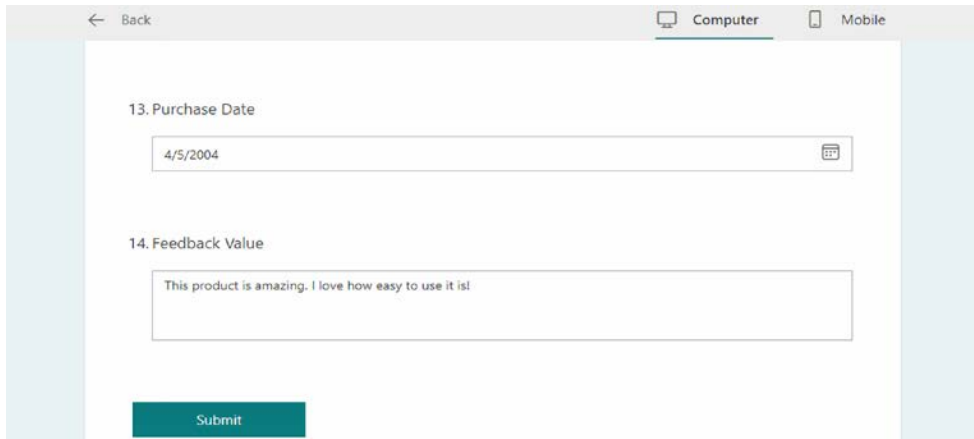


Figure 17.25: Submitting a positive test case

5. Click **Submit**.
6. Click **Submit another response**.
7. Repeat *step 4*, but substitute new test customer information. When you get to the **Feedback Value** text area, enter a value that you think reflects dissatisfaction or negative sentiment.
8. Click **Submit**.

# Reviewing the output

The easiest way to verify the output is to return to the flow test window after one of the form submissions and begin expanding the outputs. The most informational output will be located in the **Analyze positive or negative sentiment in text** action. Expand the **Apply to each** function, and then expand the **Analyze positive or negative sentiment in text** action. Review the outputs:

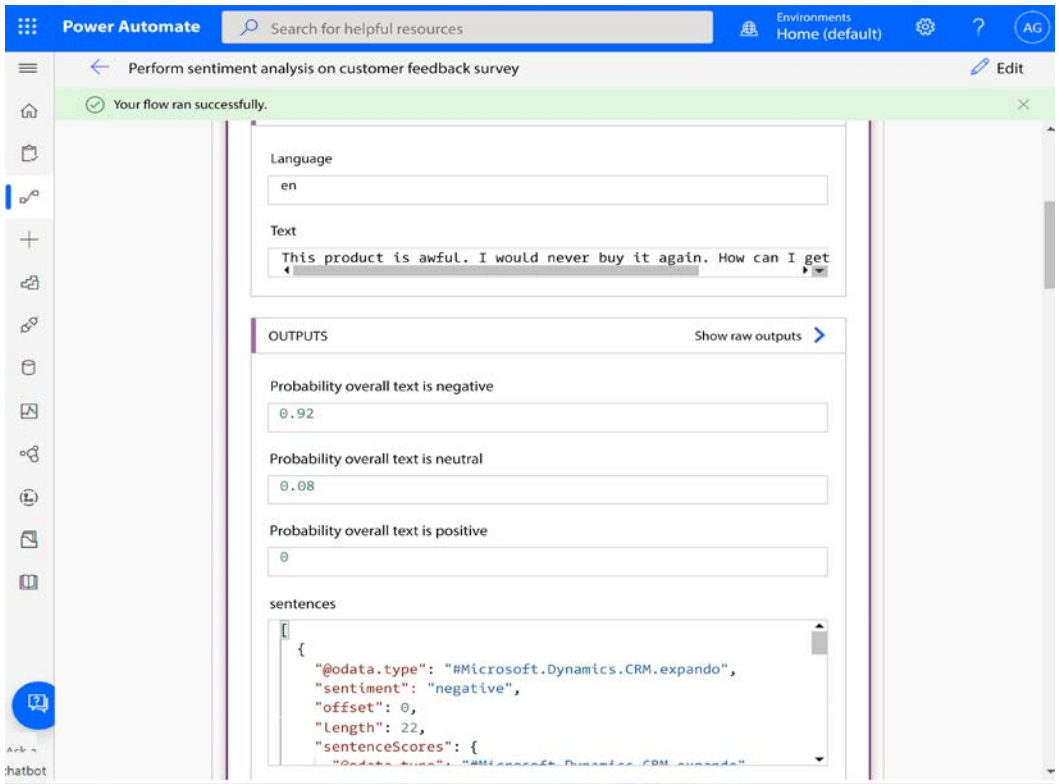


Figure 17.26: Reviewing the sentiment analysis

In the **OUTPUTS** section found in *Figure 17.26*, you can see how the action returns its results, reporting a separate probability score for negative, neutral, and positive. The higher an individual score is, the more weight is given to that particular sentiment, as shown in *Figure 17.27*:

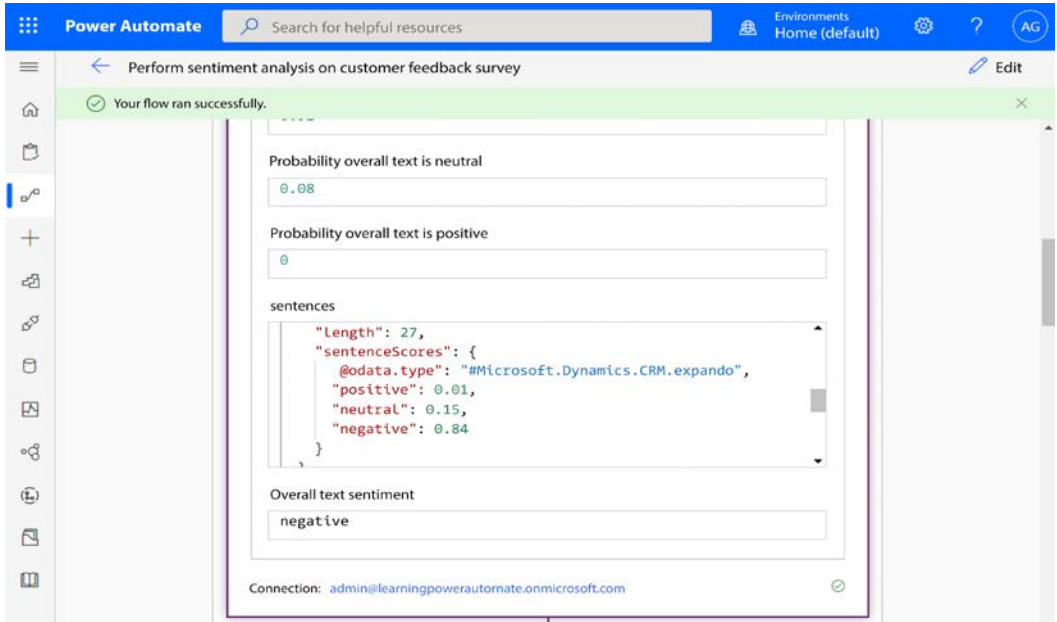


Figure 17.27: Reviewing sentiment scores

You can also open the Excel workbook that stores the data and review all the results together:

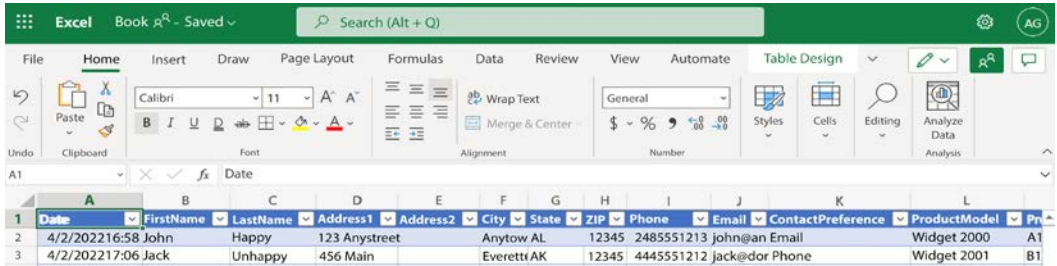


Figure 17.28: Reviewing customer survey and sentiment data

As an organization, you can work with this sentiment data to spot trends across certain products, services, geographies, and time frames to help improve customer satisfaction and experiences.

## Expanding further

Sentiment analysis is growing to be an increasingly critical part of helping organizations manage high volumes of unstructured data. Large organizations that receive a lot of feedback may desire to quickly decide if responses are positive or negative in order to focus resources on resolving potential product or service issues.

In a customer service and response scenario, it may be important to manage by exception and focus on resolving dissatisfaction. Now that you've seen how to configure sentiment analysis for a real-world situation such as processing customer feedback, think about ways that you can add to this flow to improve its usefulness. A few ideas might include:

- If sentiment analysis is negative, send an email to a customer service manager or make an entry in a contact management database for a follow-up activity.
- If sentiment analysis is positive, send the customer a promotional invite and ask them to leave a positive review on a shopping website.

To further adapt this, you might also consider creating a flow to allow you to collect customer feedback through other means – such as a dedicated feedback mailbox, Power Pages portal (also known as a Dynamics 365 or Power Apps portal), Twitter mentions or direct messages, Facebook posts, or other internal applications.

With these possibilities in mind, sentiment analysis can be an important tool for helping manage your brand's image and improve customer experiences.

## Summary

Successful businesses capitalize on capturing and responding to customers. Modern sentiment analysis allows organizations to quickly categorize customer feedback and use it to analyze trends as well as position themselves to respond to customers based on business priorities and metrics.

In this chapter, you learned how to capture customer survey data with Microsoft Forms, detect the language of the submitted data, evaluate its sentiment, and save the results to an Excel workbook for later analysis. These skills can help you identify positive and negative trends with your customers and respond to the quickly changing market landscape.

Now that we have covered some advanced flow concepts, in the next chapter we will begin to look at administrative tasks in Power Automate, starting with exporting and sharing flows with others.



## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>



# 18

## Exporting, Importing, and Distributing Flows

At some point, it may be necessary to be able to move a flow between environments or make a backup copy for archival purposes. There could be several reasons for this, including:

- Promotion from development or test environments to production
- Sharing with other teams or organizations
- Archival or backup
- Publishing to the Power Automate community

Whatever your requirements, Power Automate gives you the ability to back up or export, import, and share your flows.

In this chapter, we're going to explore the following concepts:

- Exporting a flow
- Importing a flow
- Distributing a flow

By the end of this chapter, you should be comfortable with the processes for exporting (or backing up), restoring (or importing), and sharing flows that others can then import.


### Exporting a flow

If you want to back up a flow or make a flow available for use in another environment, you'll need to export it. Exporting a flow gives you the ability to move it.

A flow can be exported into different formats:

- A package ( .zip for importing into Power Automate)
- A Logic Apps template ( .json for importing into Logic Apps)

The export format you choose depends on what the target environment is and when your flow was built. The formats are very similar, as Power Automate is built with the same technology as the Logic Apps platform. In this example, we'll look at the steps for exporting a template for reuse in Power Automate (in either the same or different environments).



The Export to Logic Apps feature is not available for Power Automate flows created after August 2020, as Microsoft changed the protocol used to create flows to the OpenAPI 2.0 standard. Logic Apps is incompatible with this protocol, so the Export to Logic Apps feature has been disabled for all flows created after August 2020.

The export dialog also allows you to retrieve the object ID (called the **flow identifier**), which you can use for other invocation methods, such as running a flow from a button in a SharePoint column.

To export a flow, follow these steps:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **My flows**.
2. Click the ellipsis next to the flow you wish to export, point to **Export**, and then select **Package (.zip)**:

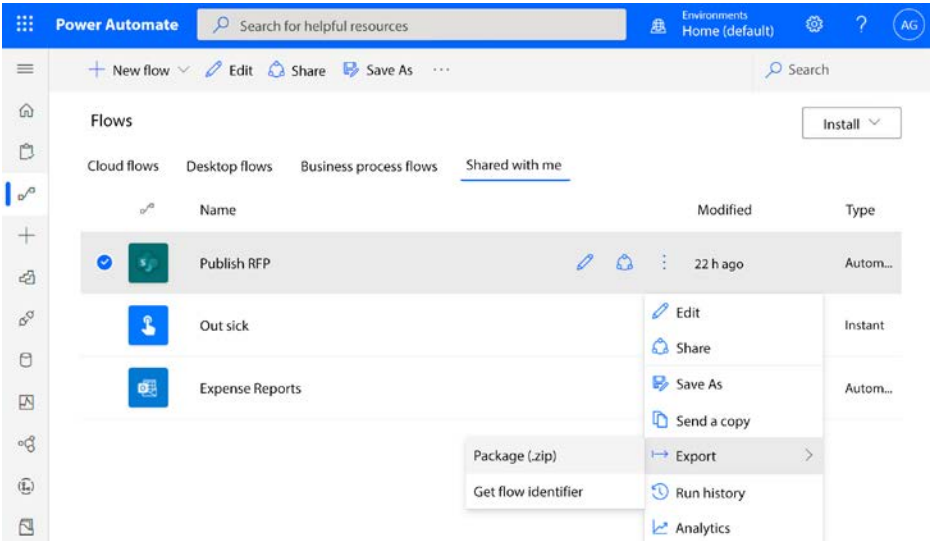


Figure 18.1: Exporting a flow in Package (.zip) format

3. On the **Export package** page, enter a name for the package. You can add comments to each of the resource types by clicking on the corresponding speech bubble icon in each resource row.

The screenshot shows the 'Export package' page in Power Automate. The top navigation bar includes 'Power Automate', a search bar, and user information. The left sidebar contains navigation icons. The main content area is titled 'Export package' and contains the following sections:

- Package details**  
Created by N/A on 03/12/2022
- Name**: A text input field containing 'Publish RFP'.
- Environment**: A dropdown menu showing 'Default'.
- Description**: A text area containing 'Publish an RFP to a public site'.
- Review Package Content**  
Choose your export options and add comments to provide instruction or add version notes.

NAME	RESOURCE TYPE	IMPORT SETUP
Publish RFP	Flow	Update

At the bottom of the table, there are icons for a key and a speech bubble.

Figure 18.2: Entering package details

- 4. In the **IMPORT SETUP** column for the **Flow** resource type, click the word **Update** to set the default for how to provision the package when it gets imported to the target environment. Select **Create as new** to create a new flow in the target environment or **Update** to update an existing flow in the target environment:

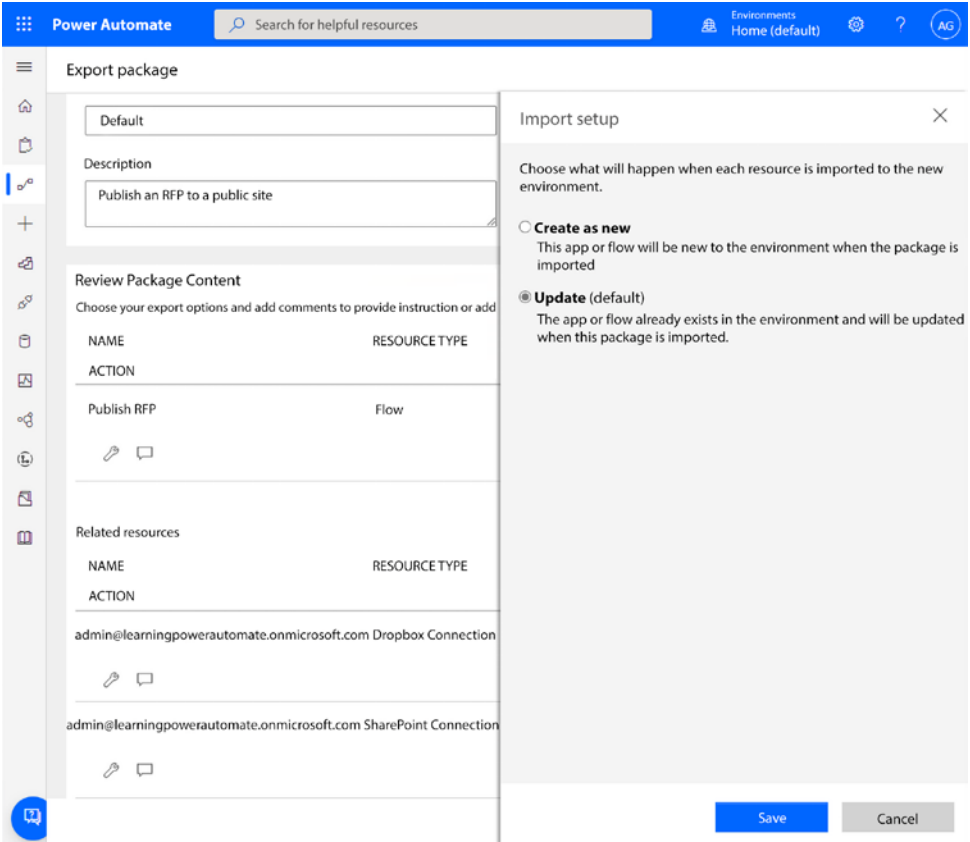


Figure 18.3: Selecting the import option

- 5. Click **Export**. Power Automate will build a .zip file and trigger the browser to initiate the download.

Once the export has finished downloading, you can distribute it as desired or store a copy for backup purposes.

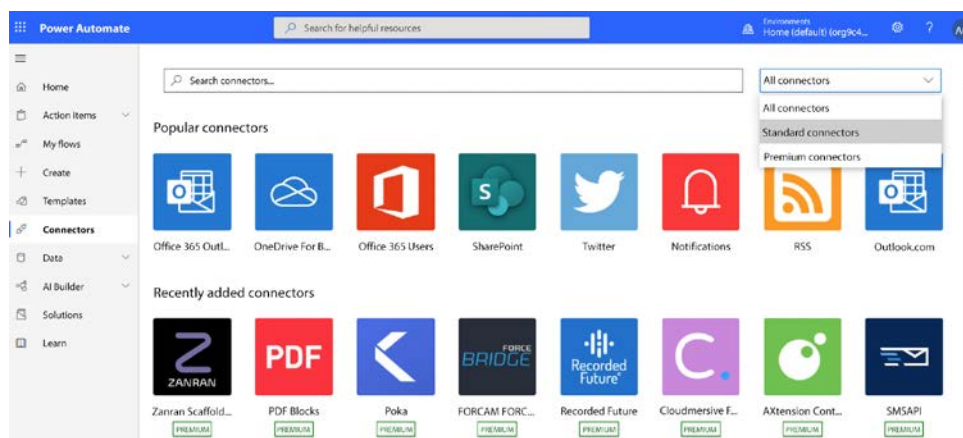
Next, we'll look at importing a flow into the environment.

## Importing a flow

Importing a flow allows you to take the exported contents from one environment and reuse the logic. The process is very simple, allowing you to move flows easily between environments.

You'll want to make sure that the flows you are importing are supported by your environment. Some examples might be:

- Connectors that may not be available in different sovereign clouds (between Office 365 Worldwide, Government Community Cloud, or 21Vianet, for example)
- A custom connector that hasn't been deployed to the target environment
- Connectors marked as **Premium** that may have additional purchase or subscription requirements in the target tenant, as shown in *Figure 18.4*:



*Figure 18.4: Reviewing connector types*

For examples and information on standard and premium connectors, refer to the following link: <https://flow.microsoft.com/en-us/connectors/>.

To import a flow, follow these steps:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **My flows**.

2. Click **Import**:

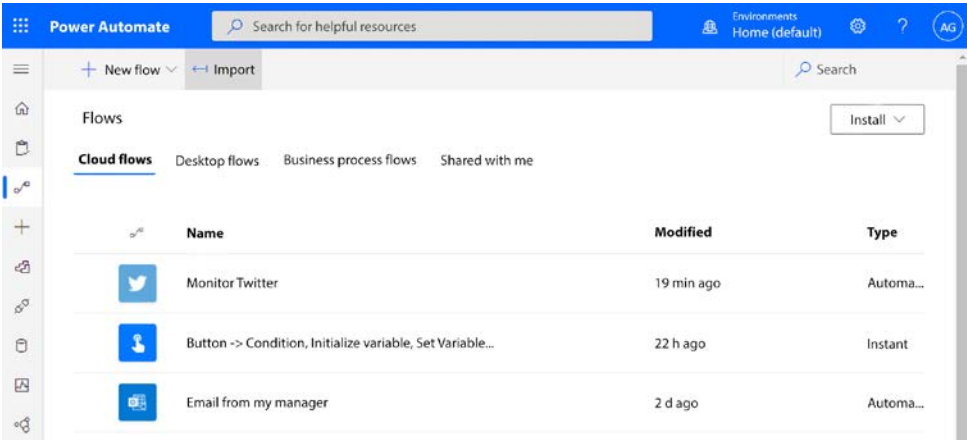


Figure 18.5: Importing to My flows

3. Click **Upload** to open a File Explorer window and browse to the .zip package file containing the flow you wish to import:

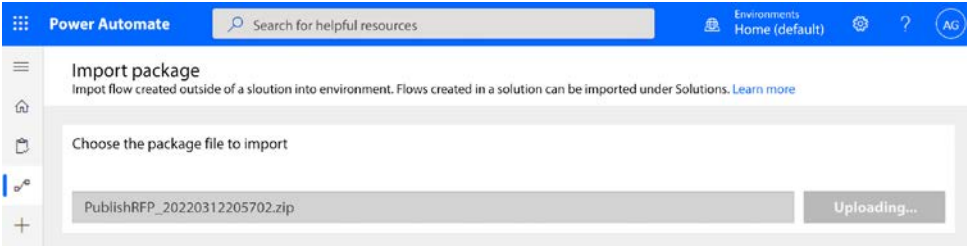


Figure 18.6: Uploading the package

- 4. Click the wrench icon under the **ACTION** column to update any items (typically credential objects) that have a red exclamation icon. Items in red represent things that need to be configured or updated in order for the import to proceed. You can also use the wrench icon to change the options that were set during the export process:

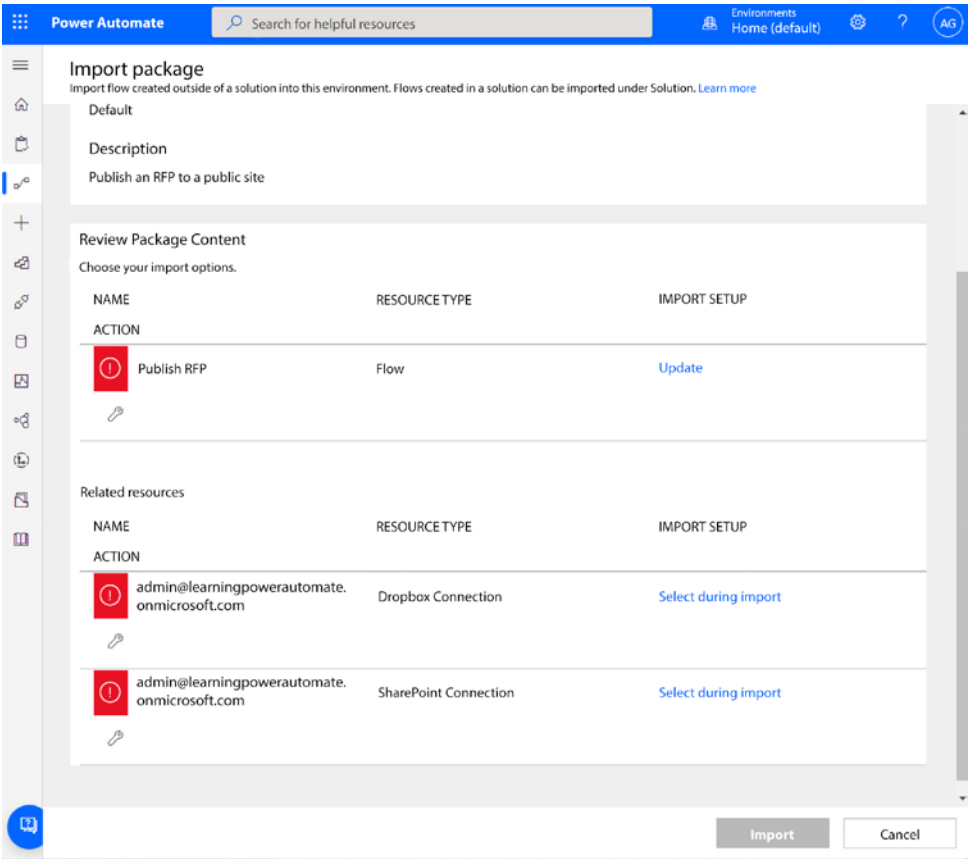


Figure 18.7: Updating items prior to import



- 5. Select an existing resource or click **Create new** to add a new resource of the correct type. Then, click **Save**:

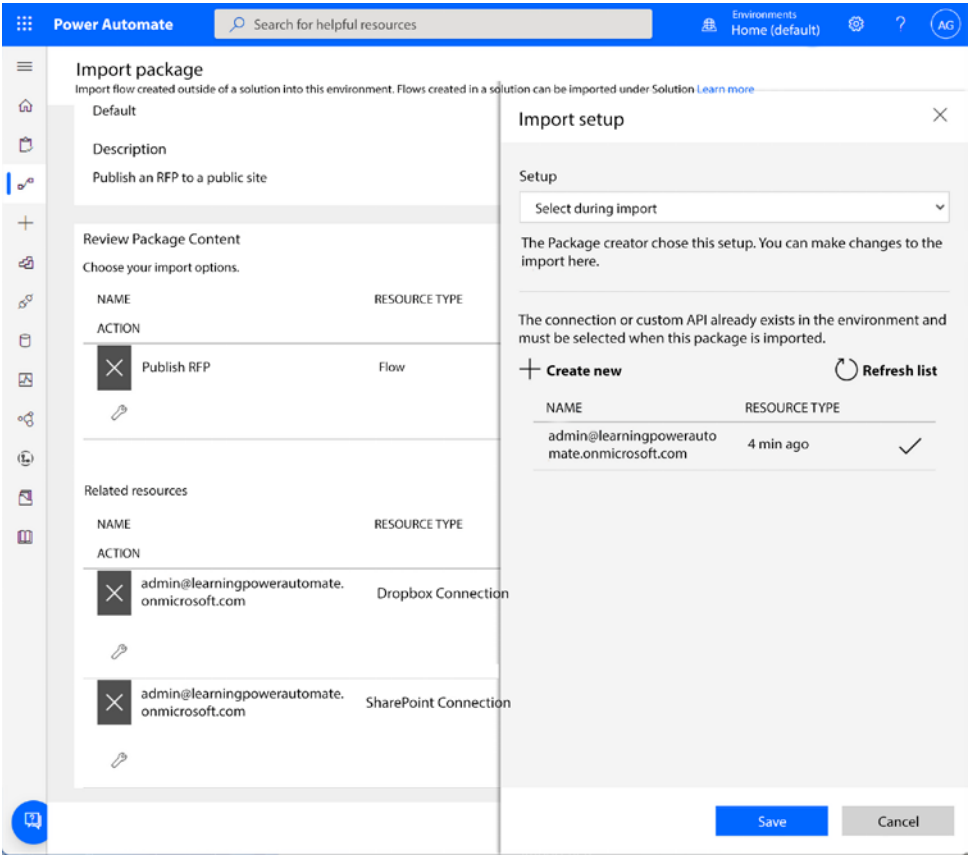


Figure 18.8: Updating a credential object prior to import

- 6. When finished, click **Import**.

The flow will be created in the environment. As with any flow template import or creation, edit it to ensure that any variables or pick lists are updated with the correct objects to ensure the flow will work as expected. If a service or resource type is not currently available in the target environment, you'll need to create it during the import step.

Missing resource or service items will be configured to **Create as new**, and you'll need to add or specify a credential for them, similar to the screenshot depicted in *Figure 18.8*. You'll need to repeat that process for every service used in a flow that isn't already configured in the target environment.

Next, we'll look at ways to distribute a flow.

## Distributing a flow

If you have exported a flow and want to share it with others, you can do so in several ways:

- By sending the exported .zip file as an email attachment
- By posting the exported .zip file on a file-hosting or sharing site such as OneDrive or Dropbox
- By sending a flow via Power Automate
- By publishing the flow as a template to the gallery

As the first two are relatively simple, we'll focus on the steps necessary to perform the last two methods: sending a flow via Power Automate and publishing the flow as a template to the gallery. Both of these methods will allow users to download and install their own copy of the flow, as you'll see.

### Sending a flow via Power Automate

When you send a flow via Power Automate, you are allowing the recipients to download and create a copy of the flow. This is *different* from sharing a flow or converting it to a shared flow, as you learned about in *Chapter 7, Working with Shared Flows*. With a shared flow, all users have access to the *same* flow instance. When you *send* a flow, each recipient gets a copy of it that they can personalize.

To send a flow via Power Automate, follow these steps:

1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **My flows** and then select the flow you wish to send.

2. Select the ellipsis at the end of the flow, and then select the **Send a copy** option:

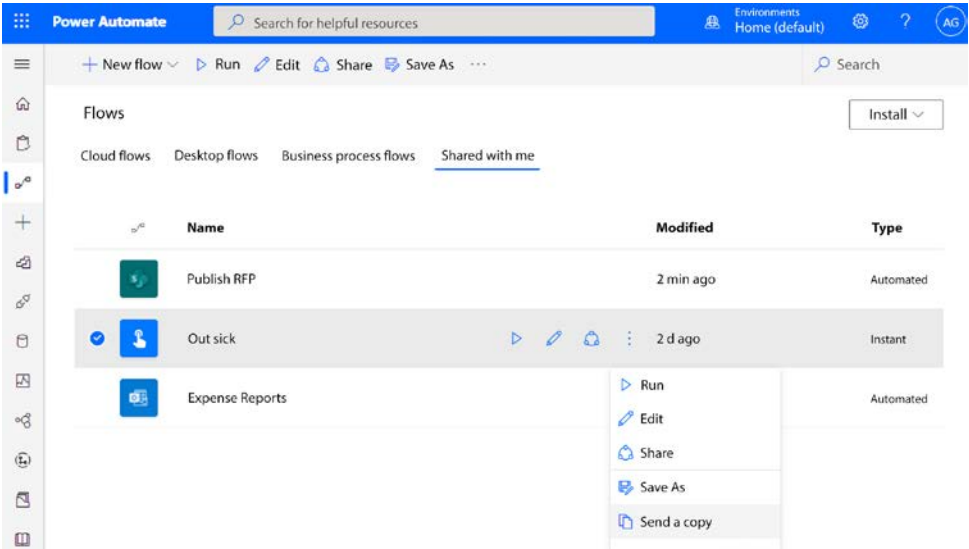


Figure 18.9: Preparing to send a copy of a flow

3. Enter the email addresses of users you wish to send the flow to, and then click **Send**:

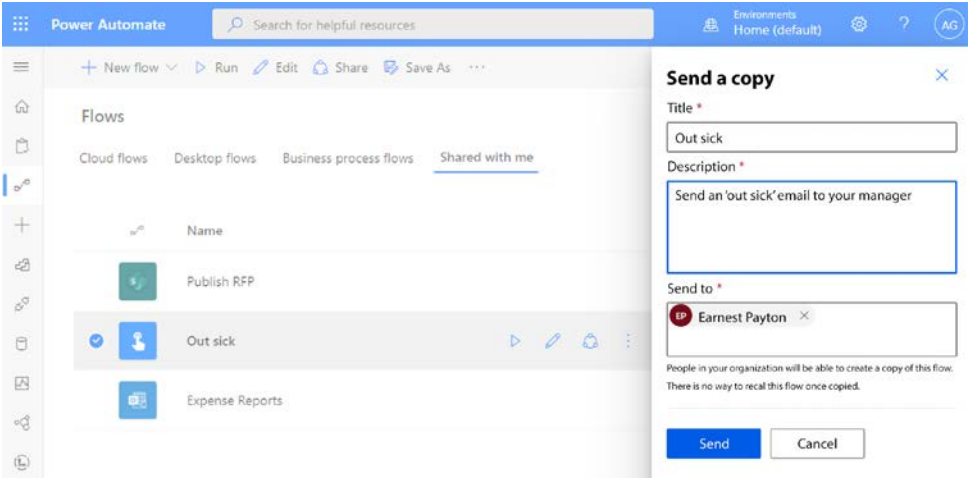


Figure 18.10: Sending the copy of the flow

4. The recipient should receive an email prompting them to add the flow to their environment.
5. If you are the recipient of a sent flow, select the **Create My Flow** button in the body of the email to start the import and configuration process:

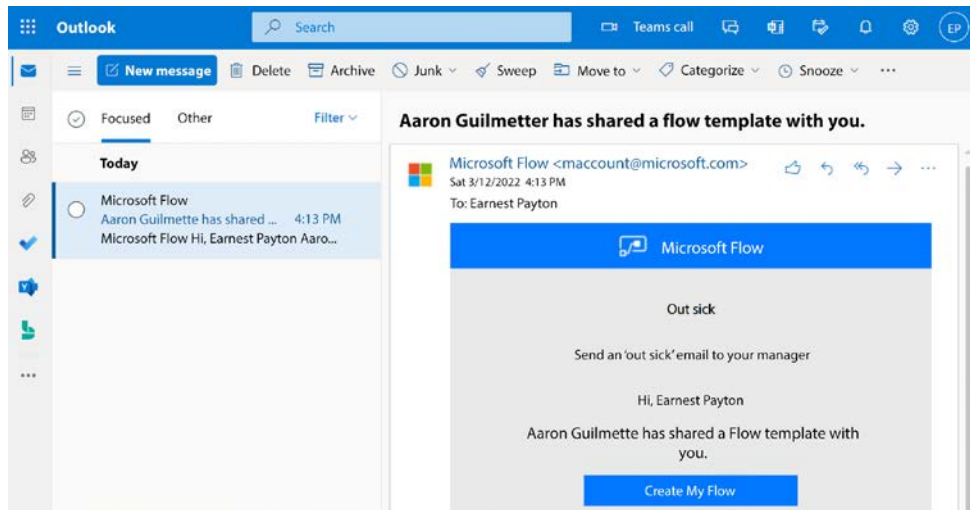


Figure 18.11: Example of email from sending a copy of a flow

6. If necessary, update any credential objects necessary to configure the flow, as shown in Figure 18.12:

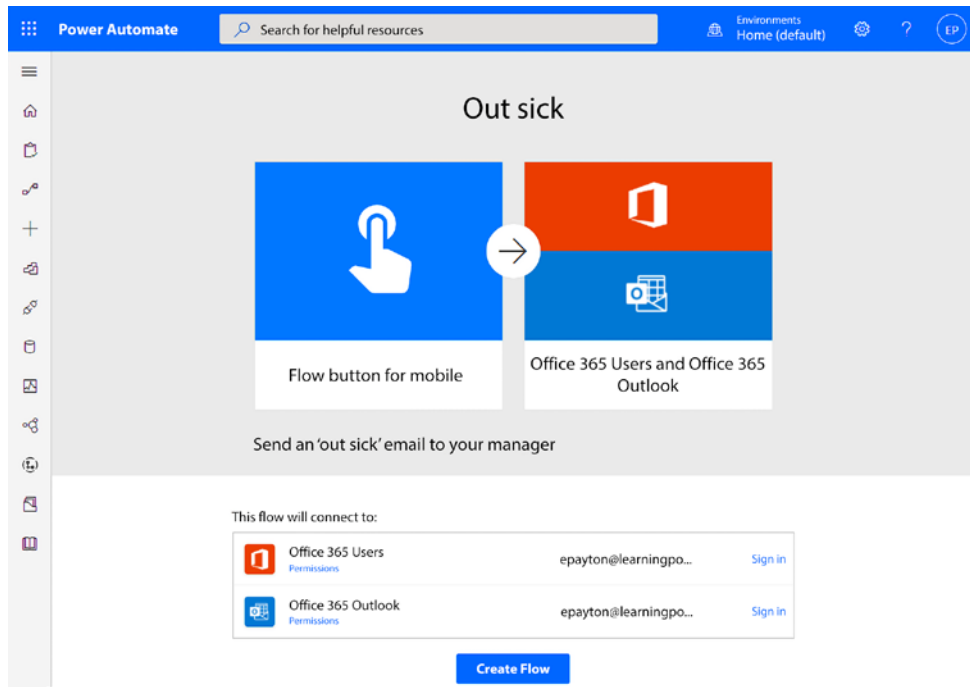


Figure 18.12: Create flow page

- 7. Update the credentials as necessary and then click **Create Flow**.

The flow will be created in the environment. As with any flow template import or creation, edit it to ensure any variables or pick lists are updated with the correct objects to ensure that the flow will work as expected.

## Publishing a flow to the template gallery

If you think your flow might help others, you can publish it in the template gallery that all Power Automate users can access. In order to submit a flow to the gallery, the most recent run *must* have been successful. This helps ensure that the flow will be successful for other users as well.

To submit a flow as a template to the gallery, follow these steps:

- 1. Log in to the Power Automate web portal (<https://flow.microsoft.com>). Click **My flows** and then click the link for the flow you wish to publish, which will open the details page.
- 2. Select the **Submit as template** option. Power Automate will evaluate the run history of your flow:

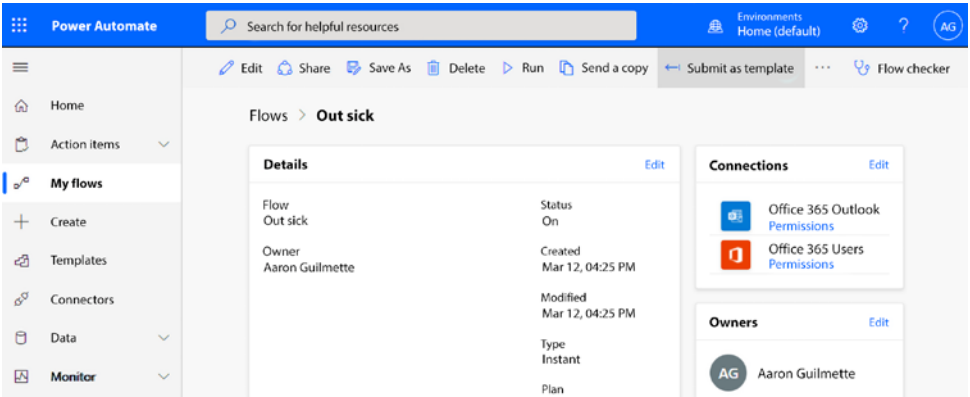


Figure 18.13: Submitting a flow as a template

3. If the run history shows the last run as a failure or that it has been modified since its last successful run, you will be unable to submit it. Correct any errors in the flow, test it, and ensure that it is successful:

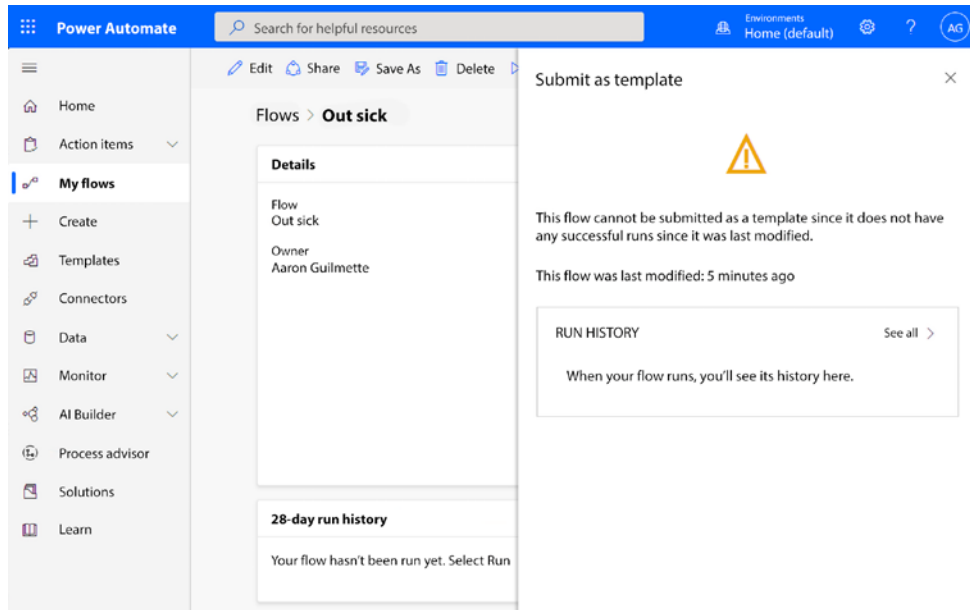


Figure 18.14: Unable to submit flow as a template

- Once the last flow run is successful, you can resubmit the flow. You'll be prompted to fill out details such as a title, a description for the template, any benefits of using the template, and more:

The screenshot shows the Power Automate interface with a 'Submit as template' dialog box open. The dialog is titled 'Submit as template' and has a close button (X) in the top right corner. The background shows the 'Out sick' flow details and its 28-day run history. The dialog form contains the following fields and options:

- Template title \***: A text box containing 'Out sick'.
- Template description \***: A text box containing 'Use this button flow to quickly send a message to your manager that you'll be taking a sick day.'
- Biggest benefit**: A text box containing 'Easy to implement; automatically sends the message to the user listed in the Manager Azure AD property.'
- Number of users**: A text box containing 'Individual'.
- Number of runs**: A text box containing 'On demand'.
- Categories**: A list of checkboxes:
  - ☐ Remote work
  - ☒ Button
  - ☐ Email
  - ☐ Mobile
  - ☐ Approval
  - ☐ Data collection
  - ☒ Events and calendar
  - ☐ Notifications

Figure 18.15: Completing the submission process

- When finished, scroll to the bottom of the panel and click **Submit**.

If accepted, your flow will be selectable from the Power Automate template gallery.

## Summary

In this chapter, you learned how to export, import, send, and publish flows using the Power Automate interface. Exporting allows you to save a copy of a flow for purposes such as archiving, sending to colleagues or external entities, or reimporting into another environment. Importing a flow allows you to take a previously exported flow and bring it into the current environment. Finally, you learned about the two ways in which you can distribute flows in Power Automate – by using either the **Send flow** or **Publish** capabilities to make the flow available to others.

In the next chapter, we'll look at some methods of monitoring and troubleshooting flows.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>







# 19

## Monitoring and Troubleshooting Flows

Despite your best efforts, at times, there will be flows that fail. It could be something as simple as an incorrect or expired credential or something more challenging to troubleshoot, such as a third-party vendor updating their interface and rendering your process invalid.

In this chapter, we're going to review some of the troubleshooting features and capabilities of Power Automate. These tips and techniques will help you troubleshoot failing flows. We'll cover the following:

- Monitoring flows
- Reviewing email error reports
- Resolving authentication errors
- Examining detailed errors with the flow checker
- Understanding error codes
- Finding additional resources

By the end of this chapter, you'll understand how to troubleshoot common errors and know where to go for additional help.

Let's go!

### Monitoring flows

Throughout this book, you've seen examples of the run history for a particular flow. In this section, we'll review the run history of a flow that is experiencing errors and troubleshoot it.

For this example, we'll use one of the purchase request approvals that we created previously, which relies upon checking for a user's manager. To experience this error, you'll need to submit the flow as a user that does not have a manager. In this example, we'll use the Procurement Two-Stage Approval from *Chapter 10, Working with Multiple Approvals*.



If you are not using a trial tenant or do not have access to an account that meets this criteria, you may need to ask an administrator to create and license a user account that meets this requirement.

## Viewing the run history

The run history will display a log of all the times that a flow has executed over the previous 28 days. To view the run history, follow these steps:

1. Using one browser session, log in to Office 365 (<https://portal.office.com>) as a user without a manager value assigned. If you are using a trial tenant to complete this process, you can simply create a new user account and choose to not assign it a manager.
2. Navigate to the procurement approval site and submit a request. If you haven't created this workflow, review *Chapter 10, Working with Multiple Approvals*, and create a similar workflow.
3. Using a second browser session, navigate to the Power Automate web portal (<https://flow.microsoft.com>) and click **My flows**.
4. Select the **Procurement Two-Stage Approval** workflow. The run history will be displayed toward the bottom of the page:

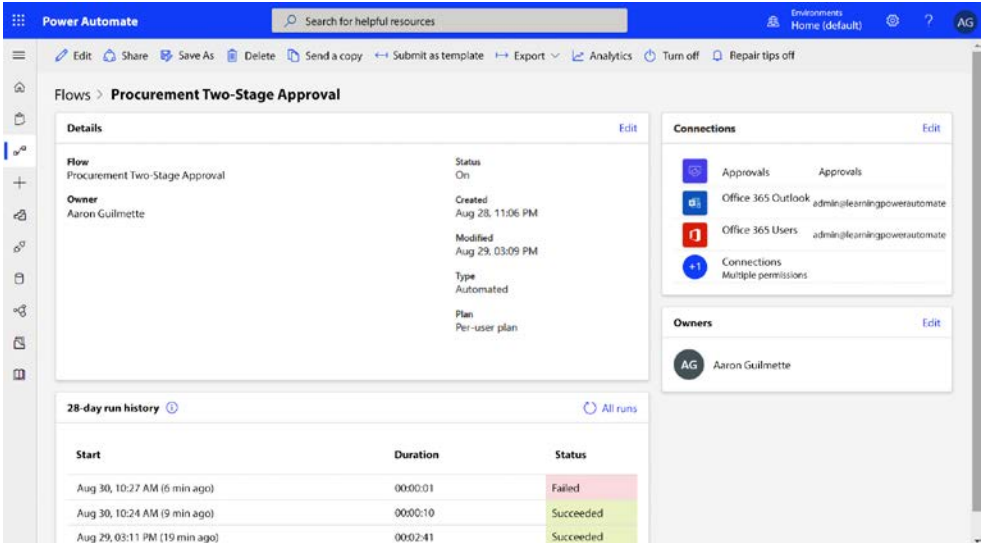


Figure 19.1: Reviewing the run history for a flow

5. Under the **Start** column in the **28-day run history** section, click the link for the date.
6. The flow run will be displayed. Errors will be shown in the fly-out panel on the right side of the page. The step in error will have a red exclamation mark on it as well:

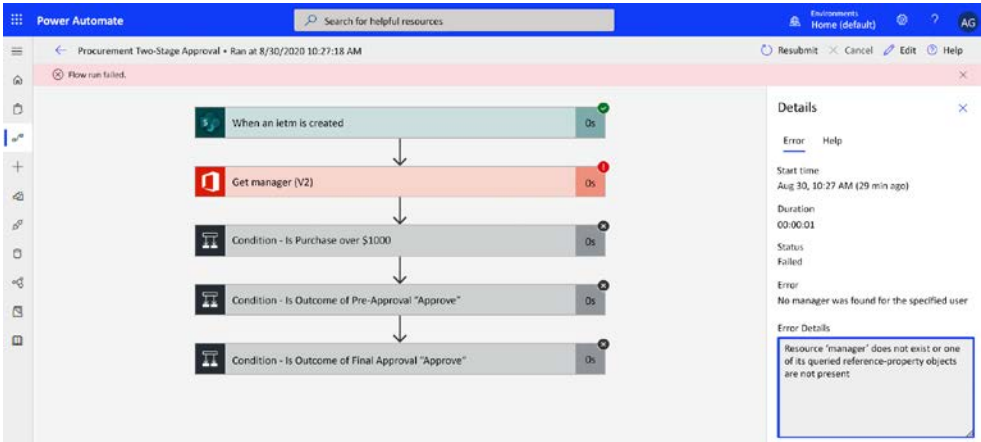


Figure 19.2: Reviewing a step error

- Most errors should have detailed information regarding the issue with the failed step. You can also click the title bar of the failed step in the main window to review the data that was evaluated during the step, as shown in *Figure 19.3*:

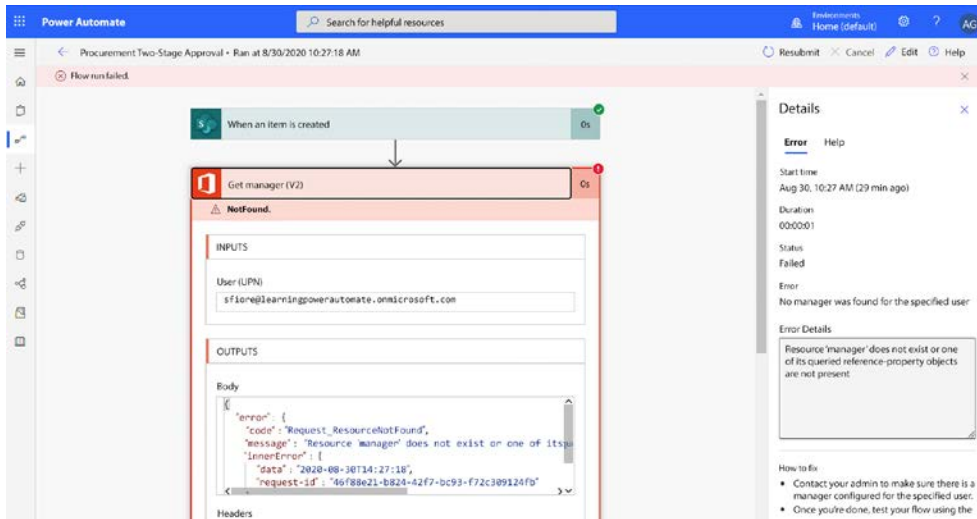


Figure 19.3: Reviewing detailed error information

Most errors provide details in easy-to-understand language, helping you clearly understand the source of the problem. Some third-party connectors, however, may have more difficult-to-understand language. You can use the error codes listed in the *Understanding error codes* section to help narrow down the cause of an issue.

Next, we'll see how to work with flows where you've resolved the issue.

## Resubmitting a flow

Once you have resolved the error (in this instance, assigning a *manager* value to the user who submitted the flow), you can have Power Automate attempt to process the flow again by resubmitting it.

To resubmit a flow, follow these steps:

- Open the failed run from the flow's run history.
- Click the **Resubmit** button:



Figure 19.4: Resubmitting a flow

3. Confirm the popup by clicking **OK**:

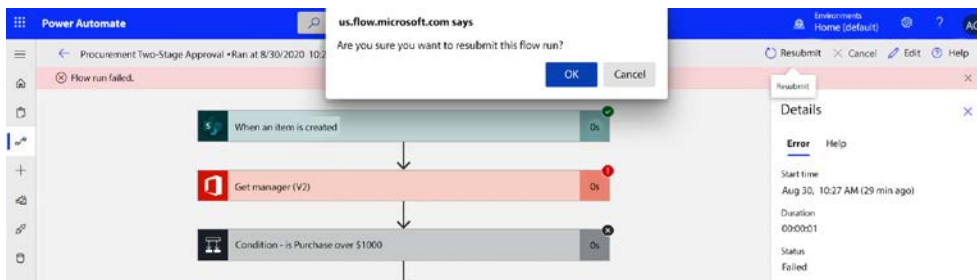


Figure 19.5: Confirming the resubmit action

4. Confirm that the flow is functioning as expected:

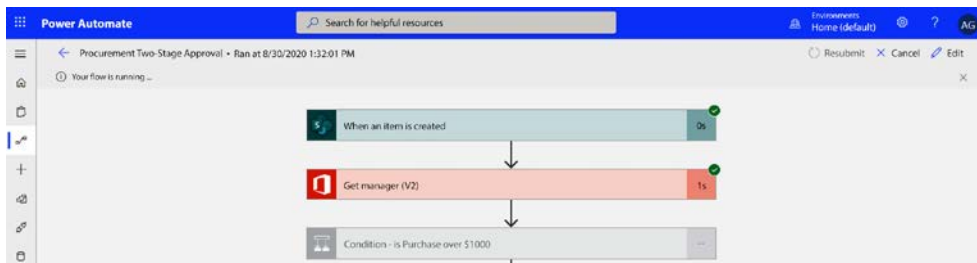


Figure 19.6: Monitoring the flow's activity

If further errors are encountered, review the run history and examine the error details.

Next, we'll look at the email error reports that are generated when a flow fails.

## Reviewing email error reports

When you are the creator or owner of a flow and it experiences an error, Power Automate will generate an email and send it to you. An example of an email error report is displayed in the following screenshot:

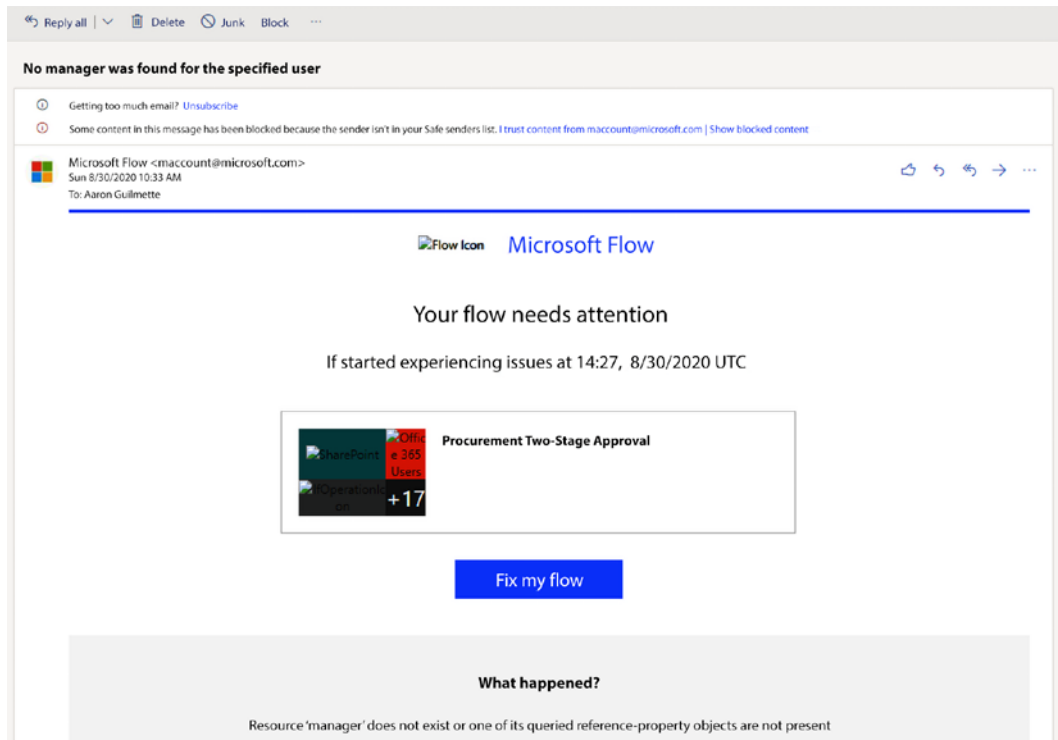


Figure 19.7: Email notification of a failed flow run

The message will contain information similar to what is displayed in the details panel of a run history error, as shown in the previous section. Clicking on the **Fix my flow** button will direct you to the **Edit your flow** page.

## Resolving authentication errors

One of the most common flow errors is due to an authentication failure. Consider the following error report email:

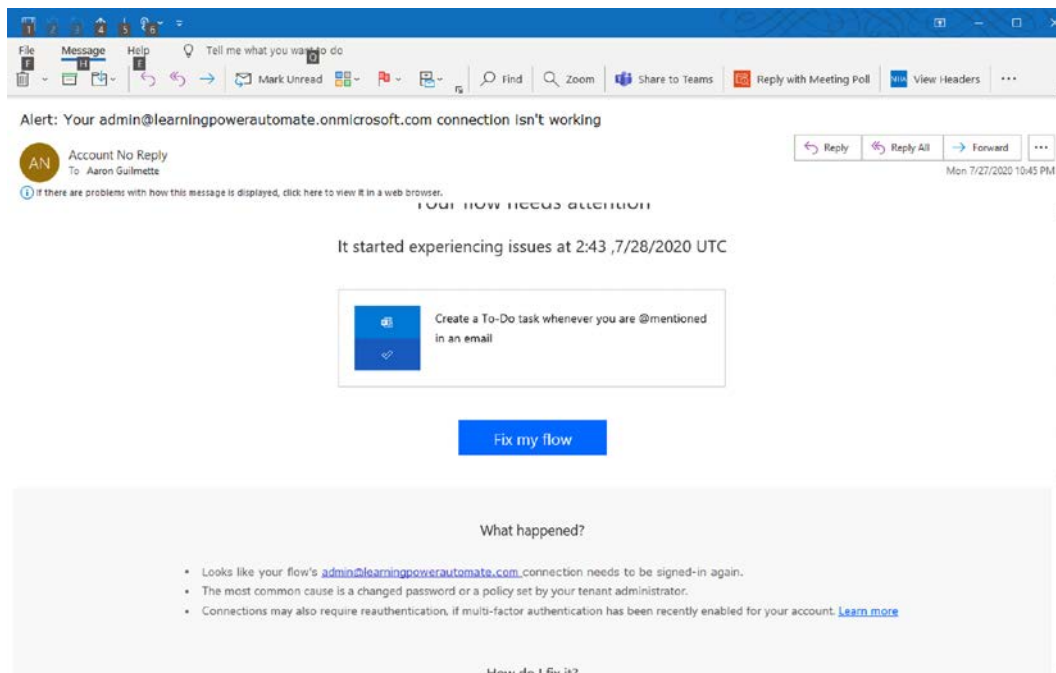


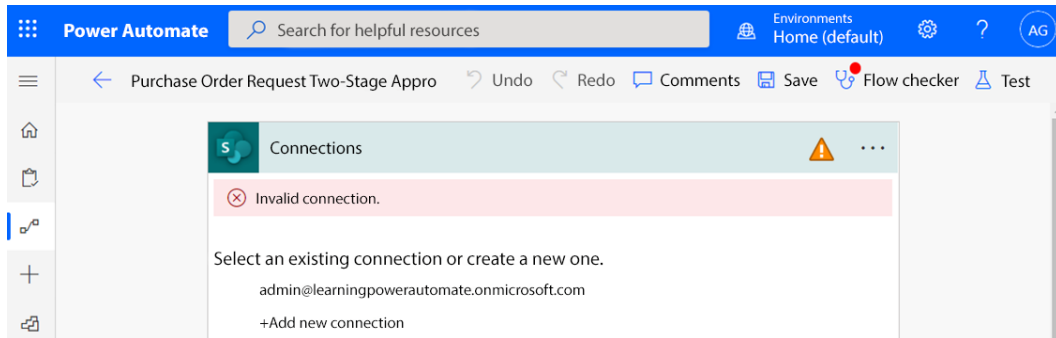
Figure 19.8: Flow authentication error

This error can occur when multifactor authentication is newly enabled for an account, your credentials have changed, or the user account used in a connection is no longer valid.

Common authentication error messages include the term **Unauthorized** or may display an error code of **401** or **403**. These error codes generally indicate that the username and password combination is incorrect or that there is some other security issue. If the username and password are correct, a 403 error code may mean that the account you're attempting to use to access a resource does not have the required permissions.



To resolve authentication errors, sign in to Power Automate, edit the flow, and then update the connection settings. Look for an action with an error and expand the title bar to update the credential being used, as shown in *Figure 19.9*:



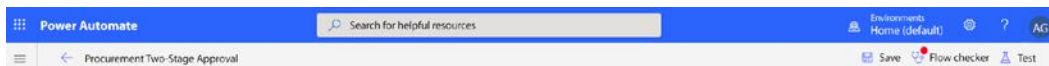
*Figure 19.9: Updating a credential object*

In the next section, we'll see how the flow checker can help you identify issues.

## Examining detailed errors with the flow checker

The flow checker is a verification and troubleshooting tool built into the Power Automate interface. It can be used during the development and troubleshooting process to help identify potential issues. The flow checker continuously evaluates your flows for potential performance, reliability, and configuration issues.

If the flow checker determines that there is a potential error, a red bubble is displayed next to the **Flow checker** icon:



*Figure 19.10: Observing a notification on the Flow checker icon*

To review the errors, warnings, or issues detected by the flow checker, click **Flow checker**. The details will be displayed in the details fly-out pane. In this example, the checker displays an error that indicates a required field has not been configured:

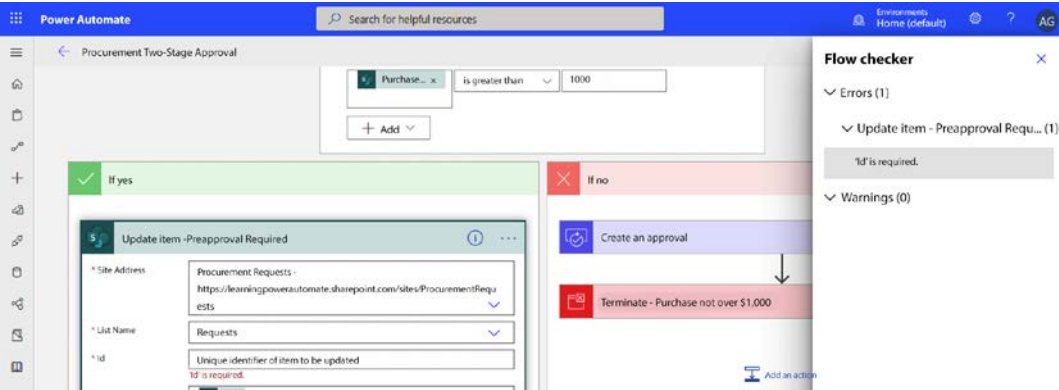


Figure 19.11: Examining an error in the flow checker

Update the items that the indicator identifies. You can then save the flow and re-run the checker by clicking on the icon.

Next, we’ll look at additional error codes that may be present when troubleshooting flows.

# Understanding error codes

You may also encounter other errors when interacting with services. Usually, Power Automate will return the actual error code it encounters when accessing a service. Here are some example error codes and some potential causes:

Error code	Details	Recommendation
401, 403	Unauthorized. This is usually due to incorrect <b>credentials</b> , but it could also indicate that an account does not have the correct permissions.	Verify that the account credentials are <b>correct</b> and that the account has access to the resource.

Error code	Details	Recommendation
400	Bad request. This error usually means that the type of data being submitted is of the incorrect type or format (such as sending CSV-type data if a field is anticipating a binary image file). It could also happen if you are attempting to send data to an incorrect URL endpoint or using an incorrect method (for example, using an HTTP GET method when you should be using an HTTP POST method).	Verify that the type of data a particular flow action requires matches the data being submitted as input. Verify that the correct URL is being used. If <b>using</b> an advanced HTTP connector, verify that the correct method (GET or POST) is being used.
404	Not found. This error means that the file or resource specified in the action is not found at the URI or URL specified.	Verify that the URI or URL paths <b>specified</b> are valid.
409	Conflict. This error may occur when two <b>processes</b> are attempting to update the same resource simultaneously.	Verify that <b>only</b> one action or process is attempting to update a given resource at a time.
423	File locked. This error can occur when <b>you</b> are attempting to access a resource that is locked for editing by another person or is checked out.	Verify that the file is not currently open by <b>another</b> user or process.
429	Rate limit exceeded. This can happen if you encounter an API interface request limit (too many <b>requests</b> in too short a period of time).	Check the flow run history to see how many runs are being executed. Check the connected service for any rate limit warnings. Reduce the number of steps or calls in the flow or run the flow less frequently.
502	Bad gateway. This can occur when you specify a URL as HTTP <b>instead</b> of HTTPS or if you are attempting to access a resource protected by a proxy server that is not responding correctly.	Verify the correct HTTP protocol is being specified in the action. If the service is hosted behind a firewall or proxy server, ensure that it is configured correctly.

Error code	Details	Recommendation
503	Service unavailable. This error indicates that the service you are attempting to reach is unresponsive, partially offline, or under heavy load.	Wait and retry. If the error persists, contact the service vendor to see whether the service is available.
504	Gateway timeout. This error can occur for services that are <b>protected</b> by a firewall or proxy and the connection times out or is reset. Additionally, this can also indicate that a service is being interrupted by an intrusion protection service.	Verify that any proxy or firewall services protecting a resource are configured correctly <b>and</b> are available and that no intrusion protection services are blocking requests.

Table 19.1: Power Automate error codes

If some of these error codes look familiar, it is due to Power Automate being built on standard web service protocols and accessing REST-based API resources. Review the errors encountered in your flow to determine whether these common errors are present.

Next, we'll review some additional resources for troubleshooting.

## Finding additional resources

There are a number of resources available to help troubleshoot and resolve Power Automate issues:

Resource	URL
Power Automate documentation	<a href="https://docs.microsoft.com/en-us/power-automate/">https://docs.microsoft.com/en-us/power-automate/</a>
Power Automate troubleshooting	<a href="https://docs.microsoft.com/en-us/troubleshoot/power-platform/power-automate/welcome-power-automate">https://docs.microsoft.com/en-us/troubleshoot/power-platform/power-automate/welcome-power-automate</a>
Power Automate community	<a href="https://powerusers.microsoft.com/t5/Microsoft-Power-Automate/ct-p/MPACommunity">https://powerusers.microsoft.com/t5/Microsoft-Power-Automate/ct-p/MPACommunity</a>
Power Automate user group	<a href="https://www.automateug.com/home">https://www.automateug.com/home</a>

Table 19.2: Additional Power Automate resources

If you're using third-party apps as part of your flows, you'll also want to review any REST API documentation that they provide. That is an invaluable resource when troubleshooting connectivity and functionality with service providers.

## Summary

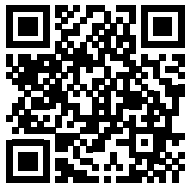
In this chapter, you learned how to use some of the troubleshooting resources available in Power Automate, including the run history and email error reports. Additionally, you learned how the flow checker can help identify potential problems in your flows. Finally, you learned about other resources available to help expand your knowledge and skills with Power Automate.

Congratulations on making it all the way to the end of the book! We hope you walk away with a greater understanding of how Power Automate can help improve your productivity, and that you remember these real-world examples of how to make Power Automate part of your toolbox.

## Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code below:

<https://packt.link/lcncdserver>





packt.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

## **Why subscribe?**

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

At [www.packt.com](http://www.packt.com), you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.



# Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



## **Microsoft Power Platform Enterprise Architecture**

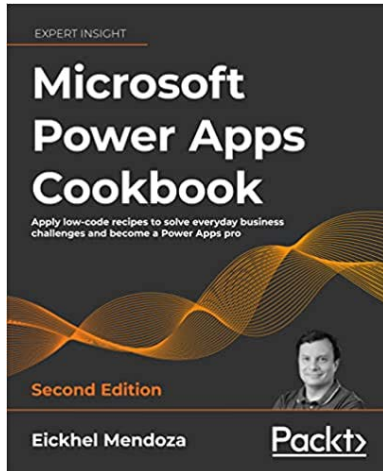
Robert Rybaric

ISBN: 9781800204577

- Understand various Dynamics 365 CRM, ERP, and AI modules for creating Power Platform solutions
- Enhance Power Platform with Microsoft 365 and Azure
- Find out which regions, staging environments, and user licensing groups need to be employed when creating enterprise solutions



- Implement sophisticated security by using various authentication and authorization techniques
- Extend Power Apps, Power BI, and Power Automate to create custom applications
- Integrate your solution with various in-house Microsoft components or third-party systems using integration patterns



### Microsoft Power Apps Cookbook, Second Edition

Eickhel Mendoza

ISBN: 9781803238029

- Learn to integrate and test canvas apps
- Design model-driven solutions using various features of Microsoft Dataverse
- Automate business processes such as triggered events, status change notifications, and approval systems with Power Automate
- Implement RPA technologies with Power Automate
- Extend your platform using maps and Mixed Reality
- Implement AI Builder's intelligent capabilities in your solutions
- Extend your business applications' capabilities using Power Apps Component Framework
- Create website experiences for users beyond the organization with Microsoft Power Pages

## Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit [authors.packtpub.com](https://authors.packtpub.com) and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

## Share your thoughts

Now you've finished *Workflow Automation with Microsoft Power Automate, Second Edition*, we'd love to hear your thoughts! If you purchased the book from Amazon, please [click here](#) to go straight to the Amazon review page for this book and share your feedback or leave a review on the site that you purchased it from.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

# Index

## A

### actions 9

- Azure AD actions 270
- database actions 211
- email actions 32
- file actions 51
- Forms actions 231
- HTTP action 270
- Power Automate for desktop action 303
- send me mobile notification action 85

### advanced conditions 119

### advanced scenarios

- everyone must approve 159
- mixed approval types 159
- working with 159

### AI Builder 7

- reference link, for licensing 338

### AI models 333, 334

- Excel workbook, creating 340-342
- Forms survey, creating 338-340
- prerequisites, configuring 338
- technical requirements 335-338
- types 334, 335

### approval flow

- approval, creating 141-149
- automated 137
- calendar events, adding 149-151
- calendar notifications, adding 149-151
- creating 137, 138
- instant 137

- SharePoint site and list, creating 138-141
- testing 152

### approval flow, configuring to use Teams 188

- approval, creating 192-197
- information, obtaining of requester 189-191
- response, returning 197-204

### approvals 7

- responding to 153-156

### Approvals connector 138

- reference link 138

### Artificial Intelligence (AI) 7

### attachments, handling in email 36, 37

- completion, verifying 44-46
- flow, creating 37-44

### attended RPA scenarios 302

### authentication errors

- resolving 379, 380

### automated flows 7

### automation

- candidates 4, 5

### Azure

- SQL server, creating 212-214

### Azure AD 269

- application, configuring 272-278
- CSV table, creating 272
- HTTP action 270, 271
- JSON, parsing 272
- prerequisites 272

### Azure portal

- reference link 244

## B

### basic form

creating 232-236

### basic sequential approval

creating 159

flow, completing 170-173

flow, creating 161

flow, testing 173, 174

prerequisites, configuring 160, 161

### Boolean 88, 127

### branching 12, 13

### business processes 5, 7

high value, high skill 4

high value, low skill 4

low value, high skill 4

low value, low skill 4

management processes 5

operating processes 5

support processes 5

### Button 7

### button flows 69, 70

creating, to email manager 71-79

executing 79-82

templates 70

## C

### client secret 272

creating 278-281

### cloud flow 7

sharing 98-101

sharing, with run only permissions 107-110

### comma-separated value (CSV) table

creating 272

### Common Data Service (CDS) 10, 135

### complex flow 186

### Component Object Model (COM) 31

### condition groups

adding 122-127

### conditions 10, 85, 115

evaluating, objects 118

example 86

expressions, using 119

multiple conditions, using 119

operators 116-118

### connection

creating, to database 220-222

### connectors 8, 53, 270

premium 8

reference link 18, 361

standard 8

### content, adding to database 222

flow, creating 223-225

flow, executing 226

flow, verifying 226

run history, reviewing for

flow verification 226, 227

SQL data, reviewing for

flow verification 227-229

### custom connectors

reference link 8

## D

### databases 209, 210

connecting to 211

connection, creating to 220-222

creating 215-217

### database table

creating 217-219

### data gateway 10, 11

### Dataverse 10, 135

entities 136, 137

**Dataverse table/entity reference**

reference link 137

**deep linking 85****design principles, REST architectures**

reference link 6

**desktop flow**

creating 102-105

sharing 102, 106, 107

**detailed errors**

examining, with flow checker 380, 381

**Dropbox**

files, publishing to 63-66

URL 64

**dynamic content 27, 303****E****email**

attachments, handling 36, 37

error reports, reviewing 378

reading 33-36

sending 46-49

working with 33

**email connector actions 32, 52**

reference link 33

**error codes**

examples 381-383

**Export to Logic Apps feature 358****expressions 53, 119****F****file connectors 52, 53****files 51**

working with 53

**files, copying to SharePoint 54-60**

results, verifying manually 62, 63

results, verifying with test flow 61, 62

**files, publishing to Dropbox 63-66**

results, verifying 66-68

**file storage mechanism 52****flow 7, 185**

components 22

creating 22-27

distributing 365

examining 29

executing 27

export formats 358

exporting 357-360

importing, into environment 361-364

monitoring 373

publishing, to template gallery 368-370

resubmitting 376

result, verifying 44, 61, 62, 66, 207, 226, 227

243, 244, 262, 298, 330, 353

run history, reviewing 243, 244

sending, via Power Automate 365-368

SQL data, reviewing 244, 245

testing 173, 204, 242, 243, 296, 328, 352

**flow, basic sequential approval**

first-stage approval, creating 162-167

second-stage approval, creating 168, 169

trigger, creating 161, 162

**flow checker**

used, for examining detailed errors 380, 381

**flow identifier 358****flow, testing**

approval, requesting 206

prerequisites 204, 205

request, approving 206

response, reviewing 207  
steps 205

## **form**

processing, with Power Automate 236-241

## **formatDateTime() expression**

reference link 78

## **Forms application**

reference link 232

## **G**

### **gateway 10**

### **Get manager (V2) action 71**

reference link 71

### **Get Manager (V2) action 95**

### **Get my profile (V2) action 71**

reference link 71

## **H**

### **HTTP action 270**

DELETE 271

GET 271

PATCH 271

POST 271

requirements 271

### **HTTP flow**

creating 281

data, gathering 281-288

output, parsing 288-290

report, creating 290-296

testing 296, 297

verifying 298

## **I**

### **input variables 303**

## **J**

### **JavaScript Object Notation (JSON) 31**

parsing 272

## **L**

### **licensing, Power Automate**

reference link 13

### **limits for automated, scheduled and instant flows**

reference link 115

## **M**

### **management processes 5**

### **Microsoft 365 clouds**

United States Government Community  
Cloud (High) 16

United States Government Community  
Cloud (Moderate) 16

Worldwide/Commercial 16

### **Microsoft 365 trial**

reference link 16

### **Microsoft Access database**

configuring 305-308

### **Microsoft Flow 2**

### **Microsoft Forms**

actions 232

connector triggers 231, 232

### **Microsoft Forms dashboard**

reference link 242

### **Microsoft Graph API 271**

reference link 271

### **Microsoft on-premises data gateway**

configuring 308-311

### **Microsoft Teams 22**

reference link 22

**Microsoft Teams connector** 186  
**multiple conditions**  
    adding, to Monitor Twitter flow 119-122

## N

**notifications connector** 85

## O

**objects**  
    evaluating, with conditions 118

**Office 365 Users connector** 71  
    reference link 71

**operating processes** 5

**output variables** 303

## P

**packages** 12

**parallel approvals**  
    working with 158

**parallel branches**  
    adding 174-183

**Peek code** 29

**Platform Notification Systems (PNSes)** 83

**Power Automate** 2, 3  
    admin 16, 21, 22  
    end user web portal 15-18  
    flow, resubmitting 377  
    flow, sending via 365-368  
    form, processing with 236-241  
    logging in 15, 16  
    mobile app 16-20  
    obtaining 13  
    reference link 236  
    run history, viewing 375, 376

**Power Automate Admin center**  
    reference link 21

**Power Automate app for Teams** 187, 188

**Power Automate Desktop** 2, 16, 20, 21  
    configuring 304

**Power Automate Glossary**  
    reference link 13

**Power Automate recorder** 304  
    running 319-323

**Power Platform** 2

**prerequisites, for Azure AD**  
    configuring 272

**Procurement Two-Stage Approval** 374

**Purchase Order approval** 185

**push notifications** 83  
    condition control 85  
    conditions 86-88  
    configuring, for emails from manager 85  
    flow, creating 88-93  
    flow, reworking 95  
    implementing 84, 85  
    reviewing 94, 95  
    send me mobile notification action 85  
    working 84

## R

**REpresentational State Transfer (REST)** 6

**Requests for Proposals (RFPs)** 63

**resources, to troubleshoot and resolve  
    Power Automate issues**  
    finding 383

**Robotic Process Automation** 2, 7

**RPA flow** 301  
    application, configuring to open 316-319  
    configuring 311



- finishing 327, 328
- framework, configuring 312-316
- prerequisites, configuring 304
- testing 328, 329
- variables, updating 323-327
- verifying 330, 331

**run history**

- viewing 374

**run only permissions**

- used, for sharing cloud flow 107-110

## S

**sample purchase order workflow** 3

**scheduled flows** 7

**schema** 272

**Selenium IDE** 301

**sentiment analysis** 355

**sentiment analysis flow**

- creating 342-351
- output, reviewing 353, 354
- testing 352

**sequential approvals**

- example 158
- working with 157

### server

- creating 211

### shared flows

- caveats 98
- connection information 98
- credential 98
- features 98
- managing 111, 112

### SharePoint

- files, copying to 54-60

### SharePoint HTTP request

- crafting, to upload data directly 263-267

### SharePoint site

### SharePoint workflows

### SQL data

- reviewing 244, 245

### SQL server

- creating, in Azure 212-214

### steps

### support processes

### switch condition

- using 127-133

## T

### team flows

### template gallery

- flow, publishing to 368-370

### templates

- reference link 11, 18

### triggers

- automated 9
- instant 9
- scheduled 9

## U

### unattended RPA scenarios

### uniform resource identifier (URI)

### user input

- options 247-250

### user input types

- flow, creating 250-258
- flow, executing 259-261
- flow execution, verifying 262, 263
- prerequisites, configuring 250-252

### User Interface (UI) flows

## **V**

**variable passing** 303

**variables** 303

    input variables 303

    output variables 303

**Visual Basic for Applications (VBA)** 31

## **W**

**workflow** 6

# Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don't worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below



<https://packt.link/free-ebook/9781803237671>

2. Submit your proof of purchase
3. That's it! We'll send your free PDF and other benefits to your email directly



